A PROPOSED SMART RESOURCE SCHEDULING SYSTEM IN FOG COMPUTING

A Thesis

Submitted to the Council of the College of Technical Engineering at Erbil Polytechnic University in Partial Fulfillment of the Requirements for the Degree of Master of Information Systems Engineering

By

Baydaa Hassan Husain
BSc. Computer Engineering

Supervised By

Asst. Prof. Dr. Shavan Askar

Erbil KURDISTAN
October 2022

DECLARATION

I declare that the Master Thesis entitled: (A Proposed Smart Resource Scheduling System in Fog Computing) is my own original work, and hereby I certify that unless stated, all work contained within this thesis is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgment is made in the text.

Signature:

Student Name: Baydaa Hassan Husain

Date: 11 - 11 - 2022

LINGUISTIC REVIEW

I, hereby, certify that this thesis, titled (A Proposed Smart Resource Scheduling System in Fog Computing) has been read and checked. After indicating all the grammatical and spelling mistakes, the thesis was given again to the candidate to make adequate corrections. After the second reading, I found that candidate corrected the indicated mistakes. Therefore, I certify that this thesis is free from mistakes.

Signature:

Name: Rozhgar Yousif Omer

Date: - 11- 2022

SUPERVISOR CERTIFICATE

This thesis has been written under my supervision and has been submitted for the award of the degree of Master of Information Systems Engineering with my approval as supervisor.

Signature:

Supervisor Name: Assist. Prof. Dr. Shavan Askar

Date: - 11- 2022

I confirm that all requirements have been fulfilled.

Signature:

Name: Dr. Roojwan Sade Hawez

Head of the Department of Information Systems Engineering

Date: - 11- 2022

I confirm that all requirements have been fulfilled.

Postgraduate Office

Signature:

Name:

Date: - 11 -2022

EXAMMING COMMITTEE CERTIFICATION

We certify that we have read this thesis: (A Proposed Smart Resource Scheduling System in Fog Computing**)** and as an examining committee examined the student (Baydaa Hassan Husain) in its content and what related to it. We approve that it meets the standards of a thesis for the degree of master in Information Systems Engineering.

Signature:
Name: Assist. Prof. Dr.Moayad Youssif
Member
Date: / 11 / 2022

Signature:
Name: Assist.Prof. Dr.Reben Kurda
Member
Date: / 11 / 2022

Signature:
Name: Assist. Prof. Dr. Shavan Askar
Supervisor
Date: / 11 / 2022

Signature:
Name: Prof. Dr.Subhi Rafeeq
Chairman
Date: / 11 / 2022

Signature:
Name: Prof. Dr. Ayad Zaki Saber
Dean of the College of Erbil Technical Engineering
Date: / 11 / 2022

DEDICATION

This thesis is entirely dedicated to my dear parents, who have always been a source of inspiration and strength for me, and who offer spiritual, emotional, and financial support on a regular basis. To my spouse, children, and friends who gave me advice and encouragement to accomplish my study.

ACKNOWLEDGMENTS

First, I would like to thank God for this opportunity and for assisting me in completing my ambition of obtaining a master's degree in Information System Engineering. Next, I want to express my thanks to the Erbil Polytechnic academic staff, all instructors during the course, and everyone who taught me at least one word.

I would like to express my deepest gratitude to my supervisor, Asst. Prof. Dr. Shavan Asker, who guided me through every stage of my thesis, from research planning to publishing.

Finally, I want to thank my family for all of their efforts to strengthen me and give me the energy to complete all of the phases of my studies.

ABSTRACT

One of the most remarkable innovative ideas in recent technology advancement is fog computing. It addresses a number of cloud computing shortcomings by bringing computation, storage, and actual services closer to end users. However, the majority of fog devices are resource restricted. As a result, without efficient resource scheduling, leveraging the benefits of fog computing is challenging.

The idea of resource scheduling is to select the most suitable resources for the applicant in order to accomplish the best scheduling target. The majority of the recent studies have been on expanding the number of fog nodes in order to improve the available resources. This has led to the emergence of a number of other problems such as increasing the cost and amount of energy consumed. Therefore, the aim of this thesis is to propose a scheduling system that will schedule the available resources in the fog layer and distribute them according to the required tasks without the need to increase these nodes. It has an extra layer, Master Fog (MF), between the cloud and general-purpose fogs, referred to as Citizen Fog (CF). The MF is responsible of task execution in CFs and the cloud. The Comparative Attributes Algorithm (CAA) is used to prioritize jobs, and the Linear Attribute Summarized Algorithm (LASA) is used to choose the most available CF with the highest computational resources. To analyze the proposed solution, iFogSim was used to create a simulation architecture and environment. The final results indicate a noticeable scheduling of available sources represented by an increase in bandwidth by 14%, in addition to an increase in processing speed by 34%. On the other hand, there was a reduction in RAM consumed by approximately 14%, in addition to a reduction in energy consumed by approximately 14%.

TABLE OF CONTANTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AMO	Ant Mating Optimization
BR	Bandwidth Ratio
CAA	Comparative Attributes Algorithm
CDC	Cloud Data Center
CF	Citizen Fog
CoT	Cloud of Things
DAG	Directed Acyclic Graph
DDQL	Double Deep Q-Learning
dEDA	deadline-aware Estimate of Distributed Algorithm
DRF	Dominating Resource Fairness
EDFD	Earliest Deadline First Dynamic
EPSO	Expanded Particle Swarm Optimization
ETR	Execution Time Ratio
FWA	Fire Works Algorithm
GA	Genetic Algorithm
GPA	Goal Programming Approach
GPS	Global Positioning System
GUI	Graphical User Interface
HEFT	Heterogeneous Earliest Finish Time
IaaS	Infrastructure as a Service
IDC	International Data Corporation
IoT	Internet of Things
LASA	Linear Attribute Summarized Algorithm
LPSO	Lyapunov framework
MF	Master Fog
MIPS	Million Instructions per Second
MOMF	Multi-Objective Moth-Flame
MOPSO	Multi-Objective Particle Swarm Optimization

MOSA	Multi-Objective Simulated Annealing
MOTS	Multi-Objective Tabu Search
NRR	Number of Requests Ratio
P R	Processor Ratio
PaaS	Platform as a Service
PSO	Particle Swarm Optimization
PSQ	Priority-Strict Queuing
QoS	Quality of Service
RR	RAM Ratio
RR	Round Robin
SaaS	Software as a Service
SDFC	Scheduling-based Dynamic Fog Computing
WFQ	Waited-Fair Queuing
XaaS	Anything as a Service

CHAPTER ONE

INTRODUCTION

1.1 OVERVIEW

As a result of significant advancement in both mobile and embedded devices, modern computing environments are becoming more diverse, with various software, hardware, and operating systems. Meanwhile, users have requested a wide range of specialized services. As a result of the increased user demands, conventional embedded systems fail to deliver higher performance (Naha et al., 2018b). Modern civilization now relies greatly on automation and the development of smart cities. Millions of new Internet of Things(IoT) products are launched every day for everything from personal usage to commercial manufacturing (Abdulqadir et al., 2021). Massive amounts of data are produced by the IoT devices that are proliferating so quickly. This data is largely comprised of big data, which must be processed using extremely powerful computing equipment. Furthermore, numerous devices need real-time services and greater precision (Aazam et al., 2018b).

The IoT scenario is supported by the computing architecture known as cloud computing, which takes full advantage of a vast and diverse variety of devices for accessing the Internet whenever and wherever it is necessary (Yousefpour et al., 2019). Enormous amount of big data are produced by the IoT devices that are proliferating so quickly. This data is largely comprised of big data, which must be processed using extremely powerful computing equipment. Furthermore, numerous devices need real-time services and greater precision (Singh et al., 2021). When IoT and cloud are combined, two significant issues will arise, each of which is unique to cloud computing. When IoT and cloud communication

occurs over a wide area network, the first issue is unacceptable response latency. As a result, the cloud-IoT architecture cannot handle applications that are sensitive to delays, which need the cloud server to respond within seconds. The delays in computation and transmission are also taken into account (Iorga et al., 2018). The second issue during integration is the significant burden on connection capacity caused by the enormous amount of data and nodes (Deng et al., 2018).

Cisco came up with the phrase "fog computing" as a solution and provided a description, noting that it refers to any device with processing, capacity, and broadband connections, that can be positioned everywhere in which there is a network connection, and that it gets the cloud sufficiently close to connected devices (Bellavista et al., 2019). The fog computing was developed as an expanded form of cloud computing to satisfy IoT requirements. The fog nodes are often dispersed close to the user, reducing latency and enabling the creation of nearby local connections (Ghobaei-Arani et al., 2020). Consequently, fog devices have a wide range of advantages over cloud computing, such as minimal connectivity, power, and processing resources. Secondly, the fog devices provide design challenges in order to achieve the performance requirements (Atlam et al., 2018).

Distributed computing infrastructure is typically used to develop fog computing. It is designed to deliver elastic resources by fetching the necessary services to a device close to the end user. Instead of being sent to the central data center infrastructure, computing is permitted here immediately at the network edge. Considering that the users and fog nodes communicate over a variety of wireless networks, including cellular and Wi-Fi networks. Additionally, the Internet connects the fog and cloud infrastructure (Mukherjee et al., 2018). To fulfill the needs of completion time and bandwidth, the original data is processed

concurrently and interim results are provided. The decision of resource processing priority when many resources requests are received is regarded as a critical problem in fog computing (Liu et al., 2018).

Even though the idea behind fog computing is intriguing, as time goes on, the amount of data and the resources needed to analyze it expand too quickly for fog computing to be useful. By 2025, data need real time will make up 30% of all created data, as predicted by the International Data Corporation (IDC) (Aazam et al., 2018a). Because of the large volume of created data and the need to carry out various processes, the resources queues for the associated fogs get overloaded with data (Varghese et al., 2020). Adding extra fog nodes to the existing nodes is a simple technique to meet this computational need. The correlational network becomes increasingly complicated and complex as the number of fog nodes expands, while also needing an excessive amount of maintenance. Increased electrical technology has a direct influence on the environment and other natural systems (Whaiduzzaman et al., 2014). If we pay great attention, we can see that not every fog node is consuming the network resources evenly. Some of unoccupied or accessible fog nodes can carry out certain duties for other fogs (Zhou et al., 2019). This thesis attempts to give a practical and effective framework that promotes the reusability of current resources in relation to the aforementioned challenge and proposing a scheduling system that schedule the available resources in the fog layer and distribute them according to the required tasks without the need to increase these nodes.

1.2 FOG COMPUTING

Researchers from Cisco system first coined the phrase "fog computing" in 2012 (Bonomi et al., 2012). The idea of processing data and application logic at the edge is not new. Around the year 2000, the idea of edge computation first surfaced. and cloudlets, a different analogous idea, were proposed in 2009 (Iorga et al., 2018). Cloud services and fog computing are both developments of a similar system that concentrates on end devices. While applications and services are deployed in wireless services, fog computing is used to connect devices such as IoT, which is consistent with the IoT infrastructure (Gonzalez et al., 2016) (Raj and Pushpa, 2022). Non-latency aware services are also supported by fog computing. If the amount of processing is not too great, it is evident that locating idle compute resources near customers would enhance overall service performance. A massive number of disparate nodes will be linked to the fog. Among these nodes are sensors and actuators (Puthal et al., 2018). When a computation is required, fog devices will do it, and storing facilities are also accessible in most fog devices for a limited period. Processing in fog will completed without the intervention of any other individuals, and is mostly done through fog processing equipment (Zou et al., 2019). A fog device can be any equipment that has computing, communications, and storage features (Ren et al., 2019).

Assume the IoT-based transportation system displayed in Figure 1.1, in which a multitude of sensors, devices, and cars become integrated into the environment. A range of sensors, like Global Positioning System (GPS) sensors, generate data. Applications in this scenario can serve with some significant real-time processing in order to establish an IoT-based transportation infrastructure. Several aspects, including speed, transportation duration, road conditions, driving patterns, traffic sensors, temperature, and others, influence the success of

the certain application goal. Because the application environment is real-time and high scalability is required, relying only on the cloud for processing is impractical. As a result, an intermediary processing and storage node, known as a fog device, is required to minimize latency. Using a variety of fog devices, it is capable of running applications near to clients with little latency. This also assists to restricted network traffic and application workload in the cloud.

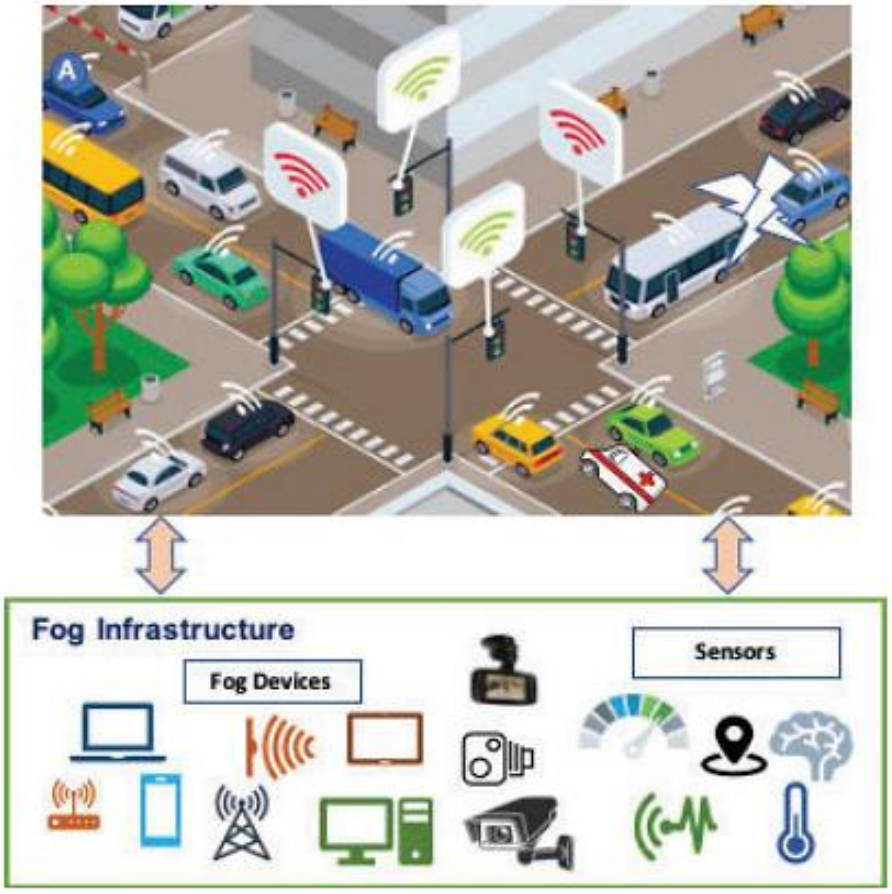

Figure 1. 1: Intelligent Traffic System Based on Fog Computing

To meet the time-sensitive needs of the applications, data analysis supplied by sensors can be conducted in the fog environment rather than the cloud. Furthermore, pushing everything to the cloud will increase network traffic

while also demand a lot of computing power; this may be accomplished using the idle computation capabilities provided in the fog computing environment's devices (Aazam et al., 2018a). Network traffic and computational overhead in the cloud are minimized by processing data given by sensors and IoT devices at the ends. To provide processing, storage, and networking services at the edge, the fog computing paradigm has been established. However, this new computer architecture raises a number of research concerns.

1.3 ISSUES IN FOG COMPUTING

The key challenges in the preceding subsection's application scenario consist of how to schedule different services in fog devices to decrease transmission delay, deal with application, and ensure the fog sustainability. Since these resources in fog devices are restricted, after scheduling is completed, assignment should be governed properly. As a consequence, in a fog environment, reliability needs may change often. Furthermore, in order to run fog applications in a live setting, reliability must be assured. User needs might change in a wide range of ways. First and initially, a user can wish to finish the prescribed task in a particular time scale; this is referred to as the deadline. Second, the user has the option of imposing financial limits. Finally, some suppliers may want to replace reality, meaning they don't actually require faultless findings but want to know how soon they can be given, even if the data contains acceptable defects (Aazam and Huh, 2015). Fog devices are self- sustainable and cannot be handled by a core application element authentication, making deployment difficult. However, because of resource constraints, cloud-based verification and rescheduling may not be optimal for the highly dynamic fog computing environment, and responding to changes will be impractical. Replication looks to be more

promising, however it necessitates the participation of several fog nodes (Al Yami and Schaefer, 2019).

In fog computing, reliability is considered as the stability of fog nodes, safe interactions, and fault tolerance. From a system aspect, it is essential for the node operating system to maintain the system's network topology, and each component in the system is capable of delivering status/diagnosis information to the edge operating system. Data reliability concerns are mainly prevalent in the data sensing and communication areas.

1.4 PROBLEM STATEMENT

It is critical to recognize that the fog is a variant of cloud computing, not a replacement. It has less processing and storage capacity than the cloud. These limited resources are one of the most important challenges facing researchers. There is a serious need to schedule the available resources for fog nodes and increase the use of it through a scheduler that ensures that resources are distributed as needed without the need to increase these nodes.

1.5 AIM AND OBJECTIVES OF THE THESIS

1. View current resource scheduling systems developed by researchers in earlier years to know the most important methods and strategies used in this field and benefit from them to plan future solutions.
2. Design a system that ensures the efficient use and distribution of resources in fog environments, taking into account the use of available resources in a scheduled form so that there are no fog nodes completely occupied and others idle. In this way we ensure that the resources are used without the need to increase the fog nodes.

1.6 RESEARCH CONTRBUTION

This thesis makes the following research contributions:

1- Developing and improving an efficient system for scheduling the available resources in fog layer.

2- Increasing the usability of the available fog resources to increase the use of available sources, resulting in increased network efficiency by reducing the need for cloud layer and contenting with the sources available in the fog layer.

1.7 THESIS LAYOUT

The remaining sections of the thesis are as follows:

Chapter 2: Describes the theoretical background and principle concepts of fog computing and the scheduling of its resources.

Chapter 3: Presents the proposal system and the used algorithms of the smart scheduler to schedule the fog resources.

Chapter 4: Describes and displays the implementation, output results, and the discussion to clarify the system represented in Chapter Three.

Chapter 5: Displays the conclusion of the thesis, and discusses the future work suggestions.

CHAPTER TWO

LITERATURE REVIEW AND THEORETICAL BACKGROUND

2.1 INTRODUCTION

Big data analytics has received a significant interest in recent years due to the fast development in the volume of data collected and the inability of standard databases to manage wide variety of data structure (Almobaideen et al., 2020). Because of the accessibility, scalability, and pay-per-use capabilities of cloud computing, enterprises now require a dynamic IT infrastructure (Bellavista et al., 2019). Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) comprise the most common cloud services, and all are on their way to Anything as a Service (XaaS) (Sadeeq et al., 2021). Further, big data, which is generated by millions of sensors, cannot be delivered and analyzed on the cloud. Furthermore, many implementations need run quicker than the cloud can presently support (Xiao et al., 2019). The fog virtualization technology is capable of alleviating the cloud's slow response problem by using the idle resources of several devices surrounding users. Furthermore, for complicated processing, fog computing relies on the cloud. Fog computing is a decentralized main technology in which the numerous devices around us with computational capabilities are leveraged (Gope and security, 2019). Even limited cellphones increasingly have computer capability, sometimes with many cores (Kumar et al., 2021, Da Silva and de Aquino Júnior, 2018). Fog computing has risen to enable applications to do processing adjacent to the user by using the idle processing capacity of disparate end and edge devices (Javadzadeh and Rahmani, 2020) (Ahmed et al., 2019). In the fog computing, the criteria and implementation of the fog environment are primary issues. Because fog

computing devices are various, the question here is by what strategies fog computing will solve the effect of interference of resource scheduling in such a shared environment (Ghobaei-Arani et al., 2020). As a result, all other relevant factors, such as deployment challenges, resource scheduling, and services, must be investigated.

2.2 OVERVIEW OF FOG COMPUTING

The Fog is a strategy for distributed computing that specifically focuses on supporting low-latency applications (Ahmed et al., 2019). Non-latency aware services can also be supported by fog computing. If the volume of processing is not too great, it is evident that leveraging idle compute resources near consumers would enhance overall service performance. The fog will be connected to a massive number of heterogeneous nodes (Naha et al., 2018a). The core compute components in the fog environment are fog devices, fog servers, and gateways. A fog device can be any device with processing, networking, and storage capabilities. Base stations, proxy servers, and other computing devices are examples of these devices. In the last few years, new obstacles in this developing computer paradigm have surfaced. This section goes through the numerous definitions of fog computing. Likewise, explore and review various fog computing research developments. Finally, examine and contrast the technological characteristics between fog and cloud computing.

2.3 FEATURES OF FOG COMPUTING

Fog computing is a cloud computing enhancement that is adjacent to things that function with IoT data. Fog computing provides a bridge between end devices and cloud computing by bringing storage, connectivity, and powerful

processors closer to user (Iorga et al., 2018). These terminal devices are known as fog nodes, and they may be deployed anyplace that has a communication link. A fog node is a tool that can operate, connect to networks, and store data. These fog nodes might be switches, servers, security cameras, or routers (Abdulkareem et al., 2019). According to (Ai et al., 2018), the core characteristics of fog computing are as follows:

I. Real-time communications: In comparison to the batch analysis used in the cloud, fog computing solicitations allow simultaneous connectivity among fog nodes.

II. Adaptability: The surrounding environment is tracked by a massive network of sensors. The fog provides storage and distributed computing resources that can support such a vast number of end devices.

III. Physical dispersion: In contrast to the integrated cloud, fog provides decentralized applications and services that may be hosted in any place.

IV. Reduced latency and increased location awareness: Because fog is close to edge devices, it reduces wait time while calculating edge device information. It also increases position flexibility by allowing fog nodes to be deployed in much geography.

V. Diversity: Fog nodes are developed by enterprises and hence represented in a range of configurations that must be managed relying on their display position. As an outcome, fog may adjust to operating technologies.

2.4 BASIC DIFFERENCE BETWEEN FOG COMPUTING AND TRADITIONAL CLOUD COMPUTING

Fog computing structures are based on fog ensembles, which are made up of numerous fog devices that cooperate to process data. Datacenters, on the contrary end, are the primary essential parts of cloud computing (Habibi et al., 2020). As a consequence, cloud computing induces large operational costs and uses a significant amount of energy. The fog computing paradigm, in comparison, uses less energy and has lower running costs. Because the fog is closer to the user, the distance between the user and the fog device may be as little as one or a few hops (Gilbert et al., 2019). Due to the distance, the cloud's interaction latency is always larger than the fogs. The cloud is a more centralized system, whereas the fog is based on geographical coordination and is more distributed (Zhang et al., 2018). The cloud cannot enable real-time interaction due to its significant latency, however this may be easily remedied using fog computing. The fog, on the contrary side, has a high failure rate as a result of wireless connectivity, scattered administration, and power failures (Caiza et al., 2020). Because fog systems will employ smart electronics and portable devices, the entirety of fog-related equipment will be wirelessly connected and are mostly distributed. Certain devices may fail if software is not handled properly. Users may be oblivious of potentially hazardous software that might cause device breakdown. Additionally, fog processing may fail in other situations, such as when each fog device is in charge of finishing its own application processing. As a result, executing IoT applications in a fog device is always comes in second. If the fog device is completely occupied by the device's application, no fog processing will proceed (Yousefpour et al., 2019) (Ghobaei-Arani et al., 2020). As a consequence, scheduling programs and resources in the fog has become more complicated. Furthermore, failure handling is competitive in the fog due to

power failure, which is only a concern when the devices are powered by batteries. Table 2.1 summarizes the technological distinctions between the cloud and the fog according to (Kumar et al., 2019) (Mukherjee et al., 2018).

Table 2. 1: Fog Computing vs. Cloud Computing

Parameters	Fog Computing	Cloud Computing
Goal	Improve the efficiency and execution of processes that should be moved to the cloud for handling, inquiry, and storage.	Provide a request for a significant improvement in the practical, powerful delivery of IT administrations.
computational emphasizes	operates at the network's edge	Data is analyzed in the cloud.
scalability	High	High
Multitask	Yes	Yes
Run time	Real time	Real time
Transmission	Device to device	Device to Cloud
Ownership	Multiple	Single
Resource Scheduling	Centralized	Centralized/ Distributed
Response Time	Fast	slow
Deployment cost	Low as it allows for deployment with or without prior preparation.	High as a result of planning and preparation
Mobility management	Easy	Hard
Reliability	Low	High
Maintenance	In most cases, no or little human intervention is required.	Technical specialists operate and maintain the system.
Size	Smaller,	Cloud data centers are enormous in size.

Resource optimization	Local	Global
Computing mode	Both a distributed and centralized fashion.	Centralized

2.5 FOG COMPUTING WITH IoT

IoT is indeed one of the prominent developments that have the ability to serve our civilization with innumerable benefits. The IoT will progress to the point where a significant number of products around us will be able to connect to Internet and interact with one another without the need for human assistance (Li et al., 2019a). Initially, the IoT designed to automate human data entry efforts by enabling various types of sensors to collect data from the surrounding and enable storage and processing of this content (Kishor et al., 2021) (Tadapaneni and Trends, 2019). Because the IoT has limited capabilities in regards to processing power and storage, It is restricted by a variety of challenges, including performance, security, privacy, and reliability (Zhang, 2020). The Cloud of Things (CoT) which is the convergence of IoT with the cloud, is the best solution to address the majority of these challenges (Hussain et al., 2019). The combination of IoT and cloud computing provides several benefits to various IoT applications. Nonetheless, because there are so many IoT devices with disparate platforms, creating new IoT applications is a challenging undertaking (Ahmed et al., 2020) (Prabavathy et al., 2018). This is due to the fact that IoT applications generate massive volumes of data. These massive amounts of data are then examined to make conclusions about various activities. Sending all of this data to the cloud necessitates an unusually large amount of network capacity (Atlam et al., 2018). Fog computing had been explored to solve these issues. Instead of transmitting data to the cloud, fog computing focuses on providing data processing and storage facilities to fog devices locally (Ngabo et al., 2021).

The intention of fog computing in collaboration with IoT is to promote performance and effectiveness while reducing the volumes of data transferred to the cloud for processing, analysis, and storage. As a result, instead of sending data to the cloud, sensors will convey it to end devices for information storing and processing, resulting in reduced network traffic and delay (Al-Khafajiy et al., 2018). Fog computing connects IoT devices with cloud computing and storage services (Zhang, 2020). To enhance the performance of IoT devices, the information can be obtained by these IoT nodes must be handled and assessed in instantaneously. Cloud networking, computation, and storage will be brought to the network's end through fog computing, solving the core problem of IoT devices while also enabling safe and efficient IoT applications (Li et al., 2019b).

Fog computing, with its distributed installations, covers a diversity of applications and services. The fog, via access points placed along extensive roadways and railways may facilitate better communication between various IoT applications, such as connected automobiles. Fog computing is regarded to be the right approach for applications needing low latency, such as multimedia applications and playing games (Shi et al., 2018) (Chalapathi et al., 2021). Furthermore, the most significant aspects of fog computing are its ability to deal with extensive sensor networks, which is a significant issue with the expanding number of IoT devices, which will soon count in the billions (Li et al., 2018). As seen in Figure 2.1, fog computing may deliver several benefits to various IoT applications.

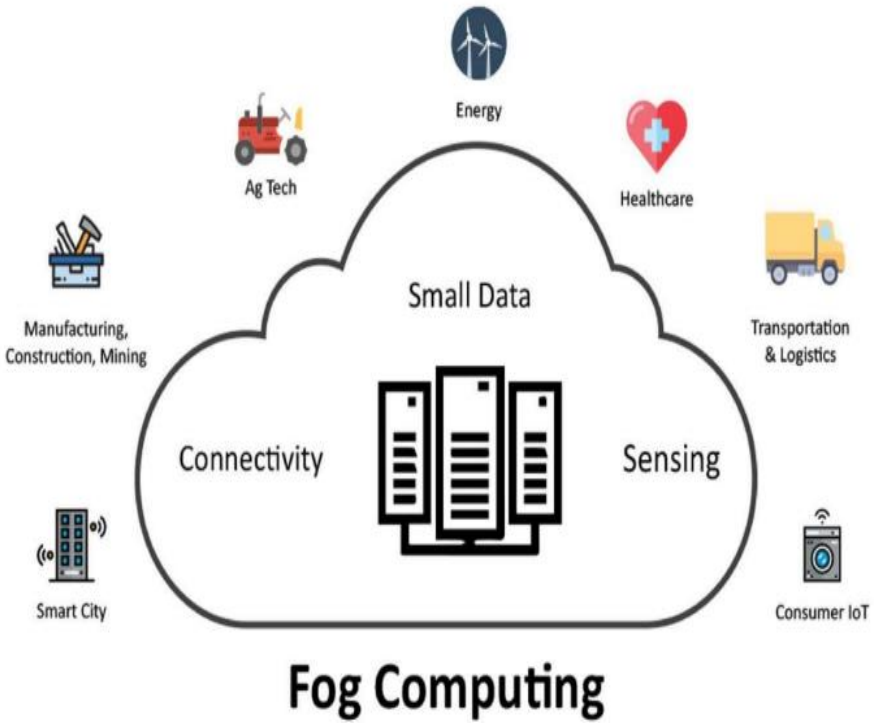

Figure 2. 1: Fog Computing Supports Many IoT Applications

Fog Computing is a comparatively modern phrase that attempts to meet the needs of applications that deal with device ubiquity. Fog serves as a bridge between the cloud and IoT, allowing them to interact. As a result, it extracts the most out of each technology, broadening the application area of cloud computing and expanding IoT resources and organizational (Al-Khafajiy et al., 2018).

2.6 THE ARCHETECTURE OF FOG COMPUTING

The Fog computing paradigm comprises three distinct levels in high-level architecture, as can be seen in Figure 2.2. The Fog layer is the most critical. This layer contains all intermediary computer devices. Traditional virtualization technologies, comparable to the cloud, can be deployed at this layer. Nevertheless, given the available resources, container-based virtualization is preferable. After processing sensor-generated data from the IoT layer, this layer

emits a request for an actuator. Although it seems that the big data matter is being handled by processing produced data at the edge level, billions of devices will create a big data quandary. At this level, modest and medium-scale big data processing is feasible. Many research works have been carried out in order to handle large amounts of data in the fog plane (Vaquero and Rodero-Merino, 2014).

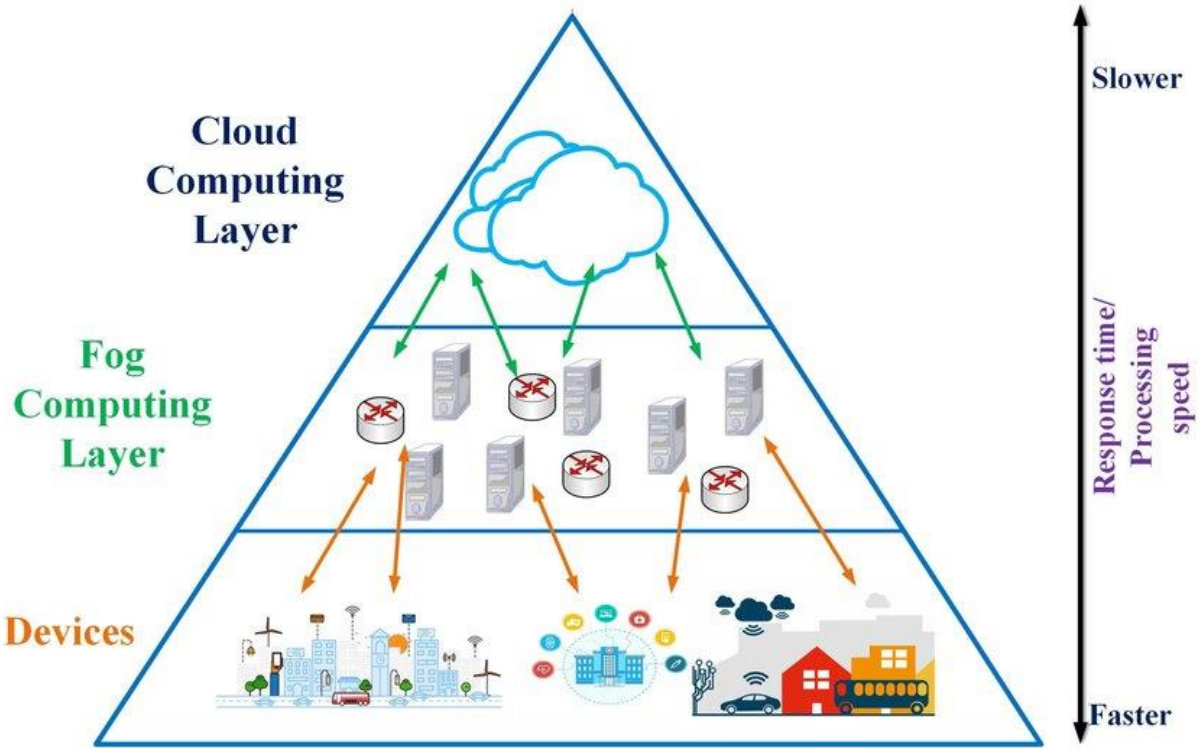

Figure 2. 2: High Level Fog Computing Architecture

IoT plane, which includes all linked devices, is the bottommost layer. The sensing and actuation processes are carried out by the devices on this plane. Processing should be done only on the fog plane for time-sensitive applications, whereas the cloud can do non-time-sensitive processing. The fog layer, on the other hand, will govern what should and should not be transferred to the cloud. Users can acquire resources from both the fog and the cloud based on their

demands. The cloud plane, on the other hand, will handle complicated processing and storage.

2.7 RESOURCE SCHEDULING IN FOG COMPUTING

Since the premise of fog computing is fascinating, the data that is generated and the resources that are required to manage it are too rapid for fog computing to be viable. According to IDC, by 2025, statistical data will account for 30% of all collected data. As a result, it emphasizes the importance of a different approach (Matrouk and Alatoun, 2021). The basic goal of resource scheduling is to pick the best resources for the client in order to achieve the desired scheduling purpose, such as enhancing resource utilization and minimizing processing delays (Li et al., 2019a).

The efficient use of computing resources is critical and necessitates an optimal and intelligent scheduling technique. Efficient resource scheduling will increase network system performance, and this scheduling is required for computer system performance optimization (Samie et al., 2019). IoT devices lack sufficient resources to store or analyze generated data, and studies show that certain connected devices lack the ability to evaluate incoming data and make choices. As a result, they require an outside entity to make choices and schedule work. Fog devices are typically network devices that have additional storage and computational capability. However, they have far fewer resources than cloud servers. As a result, efficient scheduling of these resources is critical for the fog environment to function properly. There is currently a great deal of study on resource scheduling for cloud computing at the moment, however these fog computing research subjects are still in their beginning stages (Yin et al., 2018). Resource scheduling is a decision-making challenge that is classified as a

complicated problem. Because of the diversity, complexity, and uncertainty of today's networks, creating an effective scheduler to cope with complicated problems in many network situations is a difficult task.

2.8 RELATED WORK

This section investigates a number of architecture diagram systems that are used to schedule the fog layer's restricted resources. The thoughts presented are based on recent research on these topics. On the other hand, each of the relevant works was based on one of the technologies or examined their functionality, and tested a standardized IoT scenario that integrates the features of both technologies in a single optimized application.

1. The work in (Ye et al., 2018) introduces the DeepRM2 online resource scheduling algorithm and the DeepRM Off offline resource scheduling algorithm. They presented strategies that have higher convergence speed and greater scheduling efficiency when compared to the DRL algorithm DeepRM and heuristic algorithms in terms of average slowdown time, task completion time, and awards.
2. The work in (Sun et al., 2018) proposed a two-level resource scheduling approach based on the enhanced non-dominated sortation genetic algorithm theory; they construct a resource scheduling technique for fog devices in the same fog ensembles that takes into consideration the variety of distinct devices. The results indicate that the system may be able to successfully reduce service latency while also increasing task execution stability.

3. In the work (Wu and Wang, 2019), The resource scheduling challenge is taken into consideration in the fog system to avoid total delay of tasks and meet strict deadlines. A deadline-aware Estimate of Distributed Algorithm (dEDA) with a repair mechanism and local search is employed to determine task processing order and compute node allocation. The compared findings reveal that the suggested algorithm's answer outperforms the method without the repair step. Furthermore, on both overall latency and success rate measures, this approach outperforms the heuristic technique substantially. This approach outperforms the existing fog computing resource scheduling algorithm in the majority of cases.
4. The work in (Li et al., 2019a) suggests a resource scheduling approach. They begin by standardizing and normalizing resource properties. Second, they partition the resources using fuzzy clustering and particle swarm optimization approaches, reducing the size of the resource search. Finally, they offer an improved fuzzy clustering-based resource scheduling approach. By utilizing DRL methodologies and embracing the concept of Dominating Resource Fairness (DRF), the outcomes suggest that they can increase user satisfaction and resource scheduling efficiency.
5. The work in (Bian et al., 2019) represent a FairTS technic which is a task scheduling technique for training via real observation in order to efficiently reduce average task slowness while guaranteeing multi-resource fairness among jobs. According to simulation data, FairTS outperforms government systems in terms of task slowness and resource fairness.
6. The work in (Aburukba et al., 2020) provides a tailored implementation of the Genetic Algorithm (GA) as a heuristic technique to scheduling IoT queries in order to minimize overall latency. The GA is evaluated in a

simulation environment that takes into account the environment's dynamic character. The effectiveness of the GA strategy is analyzed and contrasted to that of the Waited-Fair Queuing (WFQ), Priority-Strict Queuing (PSQ), and Round Robin (RR) methodologies. According to the data, the chosen strategy has an ultimate latency that is 21.9% to 46.6% lower than the other techniques. The proposed technique also displayed 31% improved performance in satisfying request timeframes.

7. In work (Ahmad et al., 2020), a smart healthcare application model is created and simulated using the iFogSim simulator tool, which is utilized to test and choose the strategy for introducing a Whale Optimization swarm intelligence algorithm. Swarm intelligence is compared to numerous heuristic algorithms (RR, SJF), as well as the PSO meta-heuristic method to schedule the resources. The findings reveal that the suggested algorithm reduced average resource consumption by 4.47% and cost by 62.07% when compared to the RR and SJF algorithms, and it reduced resource consumption by 4.50% and cost by 60.91% when compared to the PSO algorithm.
8. In work (Gazori et al., 2020), a reinforcement framework model was used to propose a Double Deep Q-Learning (DDQL)-base scheduler that uses a system or network while also replays techniques to address resource scheduling challenges in fog-based IoT applications with the goal of eliminating long operational delays. According to the evaluation results, the approach surpasses various baseline algorithms in terms of service latency, computing efficiency, energy consumption, and job completion, and it also overcomes dependencies and load balancing difficulties.

9. The work in (Ghanavati et al., 2020) proposes a scheduling method for fog environments with the goal of decreasing overall system make-up time and energy usage. The proposed scheme is made up of two major components: 1) a novel optimization technique is known as Ant Mating Optimization (AMO), and 2) enhanced work delivery between fog devices in direct range. The intention is to provide a reasonable trade-off between system startup and energy consumed by FC services, as measured by end-user devices. In empirical performance tests, the proposed approach outperforms the bee life algorithm, traditional particle swarm optimization, and genetic algorithm in terms of energy usage.
10. The work in (Wang et al., 2020) designed a prototype task scheduling technique known as (I-FASC), which featured an improved firework algorithm (I-FA) established by using the fireworks' explosion radius detection mechanism. This technique appears to increase task processing performance and overall dynamic routing of fog devices in a cloud-FC system, according to several pairs of research. This method can enhance the execution speed and burden in a short period of time by creating a pair of tests in this system.
11. The work in (Mutlag et al., 2020) advocates a multi-agent FC paradigm for handling medically significant activities. The fundamental objective of the multi-agent system is to connect three decision tables in order to maximize scheduling critical operations by distributing tasks based on their priority, network load, and network resource availability. The first phase is to assess if an important task can be handled locally; if not, the second step is to precisely pick the most suitable neighbor fog device to assign. If no fog device is available to process the job over the network, it is routed to the cloud, which has the highest latency. They thoroughly

assess the proposed strategy, determining its applicability and optimality at the network's edge.

12. The work in (Peixoto et al., 2021) demonstrates an element provider scheduler for a fog computing architecture with different levels of fog nodes between the edge and the cloud. The suggested scheduler seeks to meet the application's latency requirements by detecting which service components should be relocated upstream in the fog-cloud stack to mitigate computation constraint at the device level. To enforce resource access prioritization across apps, one connectivity strategy for resource distribution is suggested. The findings indicate that when applications share information across components, the proposed component-based scheduling mechanism can minimize overall delays for services and applications with high latency needs while simultaneously decreasing total network use. According to the results, this policy was able to minimize the overload impact on network utilization by around 11% when compared to the optimal allocation policy while maintaining acceptable latency for latency-sensitive applications.
13. The work in (Potu et al., 2021) developed an Enhanced Particle Swarm Optimization (EPSO) method with an additional graph technique to optimize the resource scheduling challenge in cloud-fog scenarios The basic goal was to enhance resource efficiency and lower the time necessary to conduct tasks. They evaluated the suggested EPSO method's performance to that of existing established approaches, such as ideal PSO and modified PSO; the findings showed that EPSO attained a make-span of 342.53 s. As a result, the suggested method's performance is comparable to that of previous techniques.

14. The work in (Najafizadeh et al., 2022) provides a method for securely assigning work on fog and cloud nodes based on timing constraints using Multi-Objective Simulated Annealing (MOSA). The Goal Programming Approach (GPA) is utilized to provide a feasible solution that addresses several criteria. In addition, for the deployment of IoT tasks between fog and cloud nodes, a new goal termed access level and scheduling relying on client demand is specified. The simulation findings reveal that the proposed method is 9.5% more successful in terms of service delay time, 87% more efficient in terms of access level control, and 49.8% more efficient in terms of a deadline than Multi-Objective Particle Swarm Optimization (MOPSO), Multi-Objective Tabu Search (MOTS), and Multi-Objective Moth-Flame optimization (MOMF). Furthermore, it has produced satisfying outcomes in terms of service charges that are equivalent to the average of other algorithms.
15. The work in (Yadav et al., 2022) provides a task scheduling approach that is a combination of heuristic and metaheuristic techniques. Heterogeneous Earliest Finish Time (HEFT) is a heuristic method, while the fireworks algorithm (FWA) is a metaheuristic algorithm. To decrease the number of iterations and cost variables, the bi-objective optimization strategy is proposed. The simulation results proved the importance of the provided BH-FWA algorithm above other comparable techniques. The criteria used for comparisons to prove the technique's relevance for fog computing networks include number of iterations, cost, and throughput.

CHAPTER THREE

METHODOLOGY AND SIMULATION ENVIROMENTS

2.1 INTROUCTION

With the rapid development of smart tools and other IoT, the physical connection becomes stronger with time; the number of devices connected to the network grows rapidly, and a large number of network edge devices require quick reaction time dependability. Because the Cloud Data Center (CDC) is located far away from the end user and handles a large number of IoT applications, long-distance transmission has an impact on real-time performance. Cloud computing is not a solution for all IoT issues. Long delays in IoT applications may risk the lives of patients or cause accidents in numerous critical settings, such as medical care and transportation networks. To solve these cloud computing restrictions, Cisco created a new technology called fog computing. This technology improves IoT applications by bringing computational and infrastructural requirements closer to the edge devices. As a result, services and computations may function close to terminals and IoT devices that require real-time processing capabilities with as little delay as possible. In order to offer a convincing response to the requester, the fog layer gathers and analyzes information from an end device, such as an IoT device. It also determines whether or not to send calculations to the cloud. The fog layer, like the cloud, offers end-users with data, processing, memory, and application services (Samann et al., 2021). As a result, it highlights the importance of a new approach. One easy solution for meeting this computational requirement is to expand the number of fog devices or add more processors to existing fog devices. As the number of fog devices rises, the conjoint network gets more

convoluted and complex, demanding an excessive amount of maintenance. However, if we focus our attention peeled, we can notice some idle or accessible fog devices that can perform various functions for other fogs. Some fog devices are always in use. Some fog nodes, on the other hand, may be completely available for extra task processing right now. As a consequence, depending on the conditions and time, certain fogs have a high overhead in completing a large number of operations, whilst few patches of fog are available or idle. As a result, making the most use of current resources is a desirable strategy (Ahmed et al., 2021). As a result, scheduling, executing, managing, and ensuring data security on these devices is hard, and the restricted capabilities of fog devices necessitate increased computational power. The goal of resource scheduling is to find the most valuable resources for the user in order to obtain the best scheduling outcomes, which include enhanced network throughput, reduced processing delays, and improved QoS. Scheduling encourages the allocation of resources and applications in order to achieve high throughput, responsiveness, dependability, security, lower energy usage, and competitive costs. The task scheduler assesses the devices in use to determine which tasks should be completed first. When the dynamic resource scheduling problem is solved, it will be feasible to run time-sensitive implementations in the fog environment.

2.2 SIMULATION ENVIROMENT

The fog concept was suggested to overcome the restrictions of cloud computing, where cloud servers are expanded to network edges to overcome the low responsiveness and network congestion. Several limitations must be overcome before the prospect of fog and IoT concepts for legitimate analytics can be fully achieved. The first and most significant challenge is devising

resource scheduling algorithms that choose which analysis application modules ought to be given to each end device to reduce latency and performance goals. To that end, we need an evaluation platform that will allow the consistent measurement of resource scheduling techniques on an IoT or fog computing architecture. This section outlines iFogSim, a simulator for simulating IoT and fog settings and evaluating the impact of resource scheduling approaches on latency, network congestion, energy usage, and expense.

3.2.1 The Architecture of Fog computing Environment in iFogSim

According to (Gupta et al., 2017), The layout of the fog computing environment in iFogSim is made up of various levels, with each layer responsible for certain activities that help the upper layers function. The bottommost layer of the architecture is made up of connected devices that interact with the outside environment and operate as a data source or destination. IoT sensors are selectively set as data sources for applications, detecting the environment and transmitting parameter estimates to upper layers via gateways for more processes. Similarly, IoT actuators function at the lowest layer of the architecture to regulate a mechanism. Actuators are mostly developed to respond to changes in environs specified by applications based on sensor input. Every IoT device is either a data source or a data consumer, and may thus be represented by a sensor or an actuator. In iFogSim, a sensor is paired with certain extracted features that may be altered to replicate any form of data emitting IoT device, from smart cameras to wearable environmental sensors to vehicle communication. It, like actuators, may be programmed to mimic the effects of information received from applications. Because iFogSim cannot eradicate reduced network problems such as congestion control among densely packed devices, users must abstract these

minimal difficulties to high-level characteristics such as latency or connection capacity between IoT devices and gateways. The user may construct a model of the physical level behavior of wireless characteristics of IoT nodes by detailed assessment, which can then be placed into iFogSim to imitate those affects. The framework of the fog computing environment in iFogSim is demonstrated in Figure 3.1.

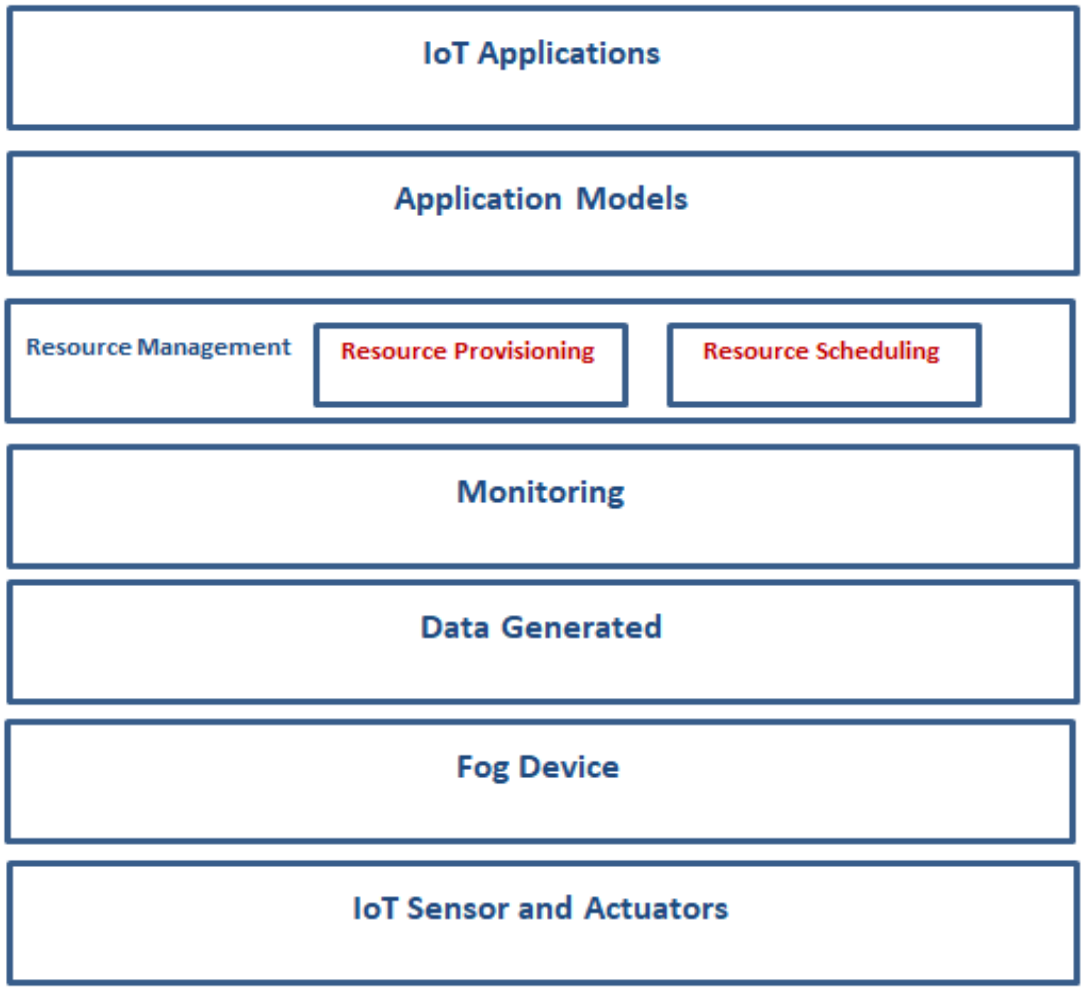

Figure 3. 1: Fog Computing Architecture in iFogSim

3.2.2 Components of iFogSim

I. IoT sensor and actuators:

A sensor is an electronic device that detects and measures physical aspects of an environment and delivers an electrical signal to a command center when certain previous events are recognized. Sensors transform physical inputs into electric signals, which are then sent to the command center. Sensors convert physical inputs into electrical outputs, whereas actuators do the inverse. Control modules translate electrical signals into physical outputs.

II. Fog device:

A fog device is any network entity that may include systems designed. Fog devices that interface sensors to the network are known as gateways. Fog devices also include on-demand cloud - based services from different geographic data centers. The fog device spans the whole resource spectrum, from these devices to the cloud. Devices are connected directly in a hierarchy. An application module executing on a fog device is in duty of processing all information created by components in the device's chain of consequence.

III. Data Streams:

The next layer of the architecture is constituted by value sets sent by the device (referred to as tuples in iFogSim). These sources can be redistributed by sensors, transported from one application module to another, or even delivered from one application module to actuators. Fog devices also provide datasets in the form of resource scheduling analytics, which are examined by the monitoring layer capable of providing perspective into device stability.

IV. Monitoring:

Monitoring keeps record of resource availability, electricity consumption, and sensor, actuator, and fog device functionality. This data is delivered

by monitoring pieces to the resource layer and, if relevant, to other services.

V. Resource management:

The most important piece of the architecture is resource management, which includes entities that manage fog device layer resources in such a way that application level QoS restrictions are achieved and resource waste is minimized. The placement and scheduler components are crucial in finding the best choices for hosting an application module and assigning the device's resources to the segment by tracking the status of available resources (information given by the monitoring service).

VI. Application models:

An application is a series of programs that make up the data processing features. Generating an estimated by one module as an output may be taken as an input by another, resulting in data dependence between the two modules. Using this application framework, we may characterize an application as a directed graph, with vertices representing application modules and controlling behaviors expressing data flow between modules.

VII. IoT Applications:

One of the main motives factor driving the growth of fog computing has been the need for actual reaction and enhanced adaptability that has accompanied the expansion of IoT. Sensors are frequently utilized as data sources in IoT applications, where the data is generally in the form of tuples. The iFogSim architecture enables two IoT application modules:

1. Model of sense-process-action: Sensor data is transferred as data streams, which are interpreted by fog device programs, and the resulting commands are communicated to actuators.

2. Model for stream processing: In the stream-processing strategy, a network of application modules running on fog devices automatically processes data streams collected by sensors. Data from inbound streams is maintained in data centers for sizable and lengthy analytics.

3.2.3 Design and Implementation in iFogSim

iFogSim's implementation is mainly composed of simulated entities and services. First, will go over how architectural elements are represented as iFogSim classes.

I. FogDevice Class:

This class identifies the specific attributes of fog devices, as well as their interactions to other fog devices, sensors, and actuators. The FogDevice class's major properties include available memory, CPU, storage space, and uplink and downlink bandwidths (defining the communication capacity of fog devices). Methods in this class define how a fog device's resources are distributed across application modules running on it, as well as how modules are deployed and terminated. Users can create custom policies for the abovementioned functions by overriding these Interfaces.

II. Sensor Class:

Instances of the sensor class are entities that operate as the IoT sensors indicated in the design. The class has properties that reflect sensor features ranging from connection to output attribute. The class includes a reference attribute to the gateway fog device to which the sensor is attached, as well as the latency of the connection between them. Most notably, it determines the sensor's output characteristics and the distribution of tuple inter

transmission, which characterizes the tuple arrival rate at the gateway. Devices can be replicated by specifying proper values for these characteristics.

III. Actuator Class:

This class represents an actuator by describing its actuation effect and network connectivity features. The class defines a method that may be altered to implement specific actuation effects when a tuple from an application module arrives. A class property relates to the gateway to which the actuator is linked, as well as the latency of that connection.

IV. Tuple Class:

Tuples are the primary unit of communication between entities in the fog and implement the architecture's data stream layer. Tuples are represented in iFogSim as instances of the tuple class. A tuple is defined by its type as well as the source and destination application modules. The class's characteristics determine the processing needs [measured as Million Instructions (MI)] as well as the length of data enclosed in the tuple.

V. Application Class:

The application design in iFogSim is represented as a directed graph, with vertices representing modules that execute processing on incoming data and edges reflecting data dependencies between modules. The following classes are used to create these entities.

1. ***AppModule***: An AppModule instance evaluates each incoming tuple and creates output tuples that are forwarded to the next modules in the Directed Acyclic Graph (DAG). The amount of output tuples per input tuple is determined by a selectivity model.
2. ***AppEdge***: An AppEdge instance represents a directed edge in the application model and signifies the data dependence between two

application modules. The kind of tuple carried by each edge is recorded by the tupleType attribute of the AppEdge class, together with the processing needs and length of data enclosed in these tuples.

3. ***AppLoop***: AppLoop is an extra class that is used to indicate the process-control loops that the user is interested in. The developer can set the control loops in iFogSim to measure the end-to-end latency. An AppLoop instance is basically a list of modules beginning with the genesis of the loop and ending with the module where the loop ends.

3.2.4 Resource Management Service in iFogSim

iFogSim features two layers of resource management for applications: placement and scheduling, which are abstracted as independent policies to permit development and adaptation.

1. Application submission: When an application is submitted, the placement policy defines how application modules are distributed across fog devices. The placement process might be guided by goals such as reducing end-to-end latency, network utilization, operating costs, or energy usage.
2. Application scheduling: The second level of resource management involves scheduling resources from the host fog device to application modules. The device's resources are divided evenly across all active application modules by the default resource scheduler. The application scheduling policy may be modified by modifying the methods in the FogDevice class.

IoT apps handle user requests and serve it based on their demands. The underlying core maintains and processes data, with the fog devices handling the majority of the work. The public cloud is in charge of processing or storage of computed data and application outputs in the fog computing platform. To handle with data at the edge, the fog layer must be equipped to handle actual data over time. The fog layer also addresses with device diversification, device interaction, device mobility, and scalability, all of which affect application performance. Various sorts of devices will be connected in the fog system. In most circumstances, the connectivity between end devices, fog devices, and fog servers will be wireless. The connection speed and throughput are affected by the kind of connection (wired or wireless). The workflow of the fog application is displayed in Figure 3.2.

The peer fog device is in charge of handling application requests and storing the results in the cloud. The fog device, IoT, fog server, and cloud all play distinct roles in the entire processing. The main processing module is the fog device. The data from IoT sensors will be processed by the fog device. The fog server will be used to collect conceptual-based relating to application processing. The fog server will connect to the cloud based on the requirements of the application. In the long-term, both the fog device and the fog server will use cloud storage to store processed data and other application-related data, but while processing, it will only utilize the information that is currently available if the others are unavailable. From here there was an urgent need for a scheduling system for the resources of the fog device.

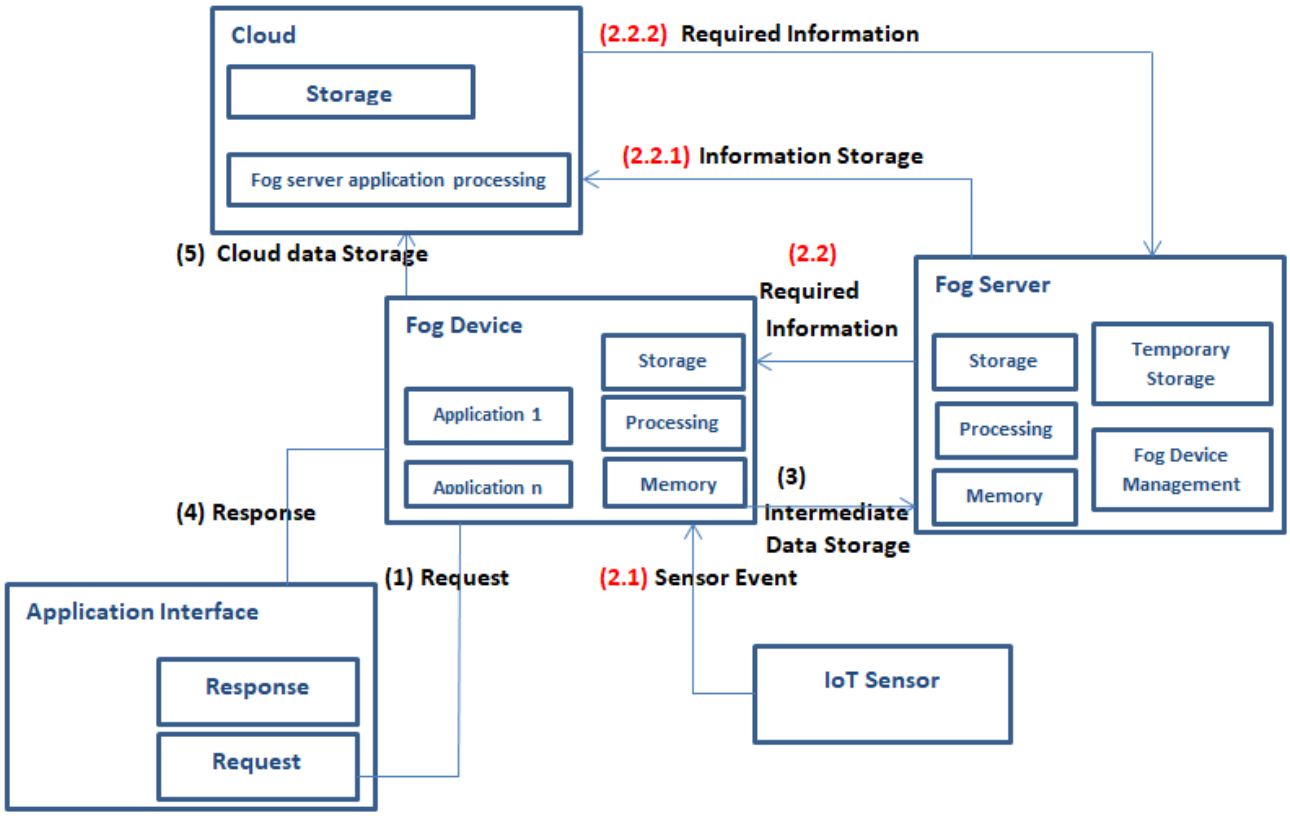

Figure 3. 2: The Workflow of the Fog Application

Scheduling manages resource usage and CPU utilization time to achieve low latency, high responsiveness, high dependability, low energy consumption, and low cost. The task scheduler examines the devices in use and determines which tasks should be executed next. This schedule makes the most use of the resources available in the fog layer to reduce the need for cloud and thus speed up the response to user requests. The scheduler will perform the reusability of available resources by using efficient algorithms.

3.3 SYSTEM DESCRIPTION

3.3.1 Basic Model

Because fog computing is a new paradigm, there are numerous problems and possibilities to meet the rising demand for resources and computation capacity. Researchers are looking to eliminate these challenges and maximize resource utilization in order to improve IoT and end-device service and performance. Researchers in **(**Hossain et al., 2021**)** proposed a system called SDFC (Scheduling-based Dynamic Fog Computing). The end-relationship user's with the first layer (general fog nodes) remains unchanged. Despite the fact that they continue to work with fog in the traditional manner, unaware of the changes in the framework, the findings suggest a better management method between the fog and cloud that can consistently deliver optimal resource usage, resulting in faster answers and decreased application failures for the users. The first layer of the system incorporates general fogs, known as Citizen Fog (CF), which are the closest nodes to the user and, when appropriate, manage communication between the cloud and the user via the Master Fog (MF). MF is the second layer which can manage request traffic that the CF cannot handle dynamically. The primary fog layer is identified. The MF establishes a settlement of a few CFs with the goal of receiving a request from one fog and allocating the duty to another fog on behalf of the asking fog. After the task is completed, this layer gets the response from the worker fog and forwards it to the asking fog. The third component of the SDFC structure is the Cloud. It is in contact with the MF. If no CF is available to execute a job, the MF sends the task and a reference to the asking CF to the cloud, and the cloud delivers the response to the referred CF straight after execution.

3.3.2 Proposed System

This section demonstrates the elements of our proposed system, which may be used to create smart scheduling in fog computing system. In our system, the SDFC primary structure was used, with a third layer of a customized fog device placed between basic fog devices and the cloud. This three-layer structure identifies and preserves the complications provided by fog and cloud. For requests accessing the MF, a context-aware queuing mechanism is used. To build a decoupled relationship between the end devices, it has been kept segregated and insulated from other fog layers. Except for the user layer, these three levels are made up of fogs and clouds. The end-relationship users with our first layer (basic fog devices) remain unchanged. Our findings suggest that improved organizational processes between the fog and the cloud, enhanced resource utilization may be attained, allowing for more efficient reactions and fewer request failures for the end-user. The critical elements of the proposed smart scheduling paradigm are depicted in Figure 3.3.

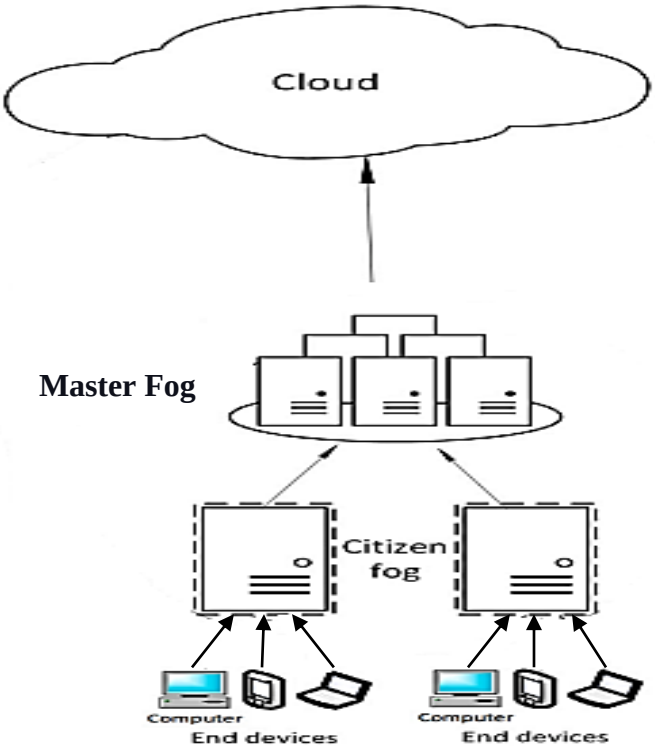

Figure 3. 3: Proposed Smart Scheduling System Paradigm

3.3.3 Proposed System Description

The smart scheduling system is made up of three layers: CF, MF, and cloud. The framework assesses the available resources for each CF and organizes actions to guarantee proper resource usage. To simulate our smart scheduling for fog resources, we use the iFogSim simulator (Gupta et al., 2017). The iFogSim simulator includes a Sensor class for producing Tuple tasks, FogDevices for processing the tasks, and an Actuator for receiving the processed tasks. We utilized Object-Oriented programming to use the FogDevice class as the CF, MF, and Cloud. Because the execution time varies based on the device configurations, we assume the

similar software environment and CF devices in the simulation. Furthermore, iFogSim produces jobs of equivalent intensity. Figure 3.4 depicts the smart scheduling system's network structure, which consists of three basic levels. The first layer from the top is the cloud layer, which is represented by the massive CDC. The second layer is the fog layer, which is made up of two sub-layers, MF and CF. Each MF is linked to a total of five CF. Depending on the density of network utilization, our suggested smart scheduling allows MF to connect to an infinite number of CFs. When compared to the SDFC system, which can only link to a maximum of 5 nodes, this is a significant benefit for our system. The CF will link to IoT devices represented by sensors or actuators, which will submit tasks to citizen nodes.

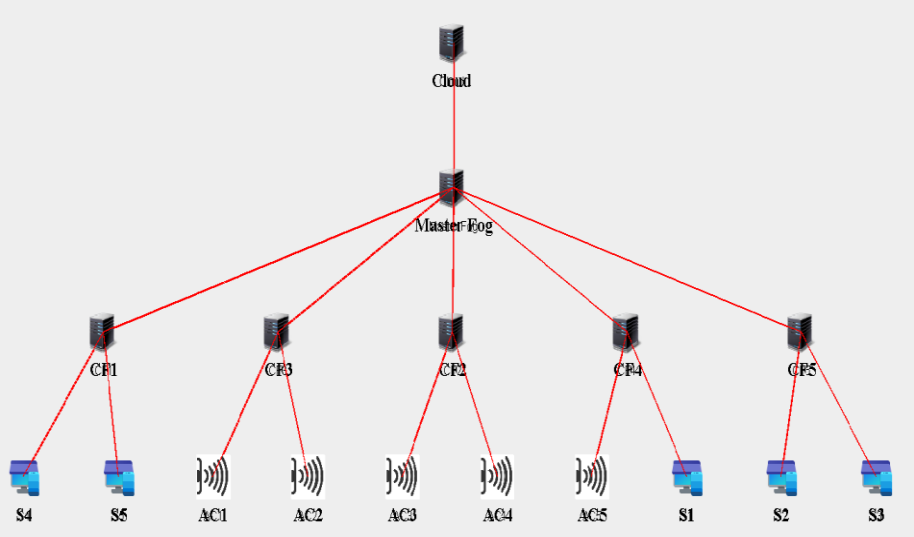

Figure 3. 4: Smart Scheduling Network Structure

1. **Citizen Fog (CF):**

   CFs makes up the initial layer of our proposed structure. These nodes are the closest to the end consumers. Smart devices, laptops, and IoT devices, for example, are linked to these fogs in the same way that any other general-purpose fog is linked. Its responsibilities include collecting tasks from users and the MF, carrying them out, and reporting the results to the respective CF. Furthermore, if it is unable to complete a task, it requests that the work be completed by the master fog.

2. **Master Fog (MF):**

   MF is the core of our approach. It communicates with both CF and the cloud. A cluster of numerous CFs is established by the MF. The density of computing requested by the end-devices determines the number of CFs utilized. The coverage zone of the MF must comprise all of the CFs. Its key responsibilities include receiving and scheduling tasks from CFs, controlling and tracking the resource status of the CFs, allocating a task to the most eligible CF or the cloud, and delivering execution results as a response to the relevant CF.

3. **Cloud:**

   Our framework's third layer is the cloud. It is assigned tasks by the MF rather than by general fogs. The MF only sends jobs to the cloud if there is no general fog and the work has been relocated to the cloud, there is no need to sustain track of the task's resource consumption. Following task completion, the cloud transmits the execution result immediately to the associated CF. Our system's primary goal is to execute actions as proximate to the end user as feasible while ensuring acceptable resource consumption in order to provide a latency aware service. To do this, we prioritize fog devices over distant cloud and low computing end-devices.

The MF was formed to help with the challenging issue of resource distribution and task management. If a CF fails to perform a task, the MF immediately assigns it to another CF. The number of resources and user tasks changes periodically to test the efficacy of our strategy. The intelligent scheduling system prioritizes jobs and filters CFs depending on their resource availability.

3.3.4 System Model and Assumption

Table 3.3 shows the abbreviations used in the mathematical model and algorithms.

Table 3. 1: The Abbreviations Used in the Algorithms and Mathematical System.

Symbols	Description
C	A citizen fog node
T_c	Execution time for c
IC	Instruction Counter
CPI	Clock Per Instruction
CT	Clock Timer
MET	Minimum Execution Time
ETRc	Execution Time Ratio of c
u,v,m	Set of tasks
n(m)	Number of common subtasks in m
Ms	Set of subtasks of m
Wu	Net workload for Wu
MW	Minimum Workload
WLR	Workload Ratio
NRR	Number of Request Ratio

MNR	Minimum Number of Requests
Ni	Number of requests of the *i*th ask
Sj	Set of depending tasks
Du	Number of tasks depends on u
Duv	Tasks v depends on u
RR	RAM Ratio
PR	Processor Ratio
BR	Bandwidth Ratio

1. Step 1: Receiving and implementing tasks at CF

We consider using fog nodes referred to as ***c*** positioned close to the users points. If we assume that each fog node receives a set of tasks from different users under the name of *u*, *v*, and *m*. Each fog node, using its algorithm, will calculate a workload for each task (Wu, Wv, Wm). In addition to calculating each subtask related to the main task. Assuming that, the set of subtasks of *m* is called *Ms*, while *Wu* represent the net workload of *u*. These measurements allow the task manager to identify the most overloaded tasks and direct them to the least busy fog node to ensure that their requests are processed at full speed. Once the process completed, it is directed to the user.

2. Step 2: Directing tasks to MF

In case that one of set task couldn't satisfied at CF, the algorithm of the specific node c will have the duty of directing these set of tasks (*u, v, m*) with all its dependence to the nearest MF. Now the MF will calculate workload for each task (*Wu, Wv, Wm*). In addition to calculating each subtask related to the main task. Assuming that, the set of subtasks of *m* is called *Ms*, while *Wu* represent the net workload of *u*. According to the algorithm, the tasks will be arranged in

preference and according to the workload. The satisfied tasks will be directed to the source CF.

3. Step 3: Directing tasks to the cloud.

Let’s consider that some of these tasks couldn’t be satisfied by either CF or MF. Then the LASA algorithm will direct that it to the cloud. Next, the LASA returning backs the received task to source MF.

The execution time of a task is determined by a CF's resources and computing ability. A CF with more resources and compute power completes a task faster than others. Aside from that, not all tasks are equally vital. In traffic system, locating an ambulance and clearing oncoming traffic for the ambulance is more critical than maintaining overall traffic flow. Furthermore, performing a higher priority job with a CF with greater resource and computing capability minimizes latency in real-time services. The Task Manager schedules tasks based on their priority, and the Decision Manager sorts CFs based on their resource capacity and performance, according to the provided algorithms. Fog capacity is the quantity of resources available for a certain activity, such as MIPS, RAM, CPU, and processor clock speed. A fog device's computing power is determined by a set of resource restrictions such as RAM, CPU, Bandwidth, and Processor Performance. We analyzed the Execution Time to run a job with the base requirement to compare the performance of a certain CF. We used these characteristics to rank the CFs based on their computational power. The average execution time was calculated using the Performance Equation. We calculate the ratio of the execution time for each CF with the Minimum Execution Time (MET) of a certain CF to compare the execution time to execute a specific job of each CF.

For a CF c, the execution time according to the Performance Equation is:

$$Tc = IC * CP * CT$$

Where,

$T_{C=}$ Execution time for c

$$CT = \frac{\text{seconds}}{\text{clocks}}$$

$$CPI = \frac{\text{clocks}}{\text{instruction}}$$

$$IC = \frac{\text{instruction}}{\text{program}}$$

The execution time Tc varies depending on the setup. However, our goal is to create a device that can perform a variety of jobs quicker than other devices, regardless of setup. As a result, we are simply evaluating the entire execution time. A gadget receives more points if it completes a particular amount of tasks faster than other devices. Now, for n number of CFs:

$$MET = \min\cup \bigcup_{i=1}^{n} Ti$$

Where $T_{i=}$ Execution time for *i*th CF.

The Execution Time Ratio ($ET\ R$) for a CF c is:

$ETR_C = \frac{Tc}{MET}$

On the other hand, not all tasks are equal in importance. As a result, we used the number of requests for each job, the interdependence of the tasks, the anticipated time to complete a task, and the burden or task-length of the assignment to prioritize and arrange the tasks. Likewise , we calculate the Number of Requests Ratio (NRR) to compare the task requests. Therefore, the Minimum Number of Requests (MNR) for *i*th task

MNR= $min \bigcup_{i=1}^{n} Ni$

Where N_i = Number of request for ith task and Ni is the number of requests of the ith task. Therefore, the Number of Requests Ratio (NRR) for ith task,

$NRR_i = \frac{Ni}{MNR}$

Where Ni = Number of request for ith task.

3.3.5 Algorithm for Scheduling Resources

Our method utilizes CAA (Comparative Attributes Algorithm) and LASA (Linear Attribute Summarized Algorithm) to ensure that the resources of certain CFs are efficiently handled. The algorithms ensure that the CF acquires the maximum amount of readily available resources and that the time spent waiting for more challenging work is minimized. If a CF is unable to finish a task, it instantly forwards it to the MF, which redistributes it to the best qualified CF. This technique is continued until all resources have been expended. The CFs not only finished their tasks, but they also finished the tasks of other CFs. The MF's whole collection of CFs acts as a much greater fog. As a consequence, the cloud is less reliant, and real-time demands are provided more swiftly.

- **CAA Algorithm**:

To define the CAA, we examined four factors: tasks Dependency, the number of tasks depending on a certain task, the Number of Request Ratio (NRR), and the Workload Ratio (WLR). When a Task Queue (TQ) gets a job, it forwards the task queue to the Task Manager (TM). The queue is prioritized by the TM depending on the aforementioned criteria. To do this, the TM employs a Divide and Conquer strategy. Divide and conquer segments the queue incrementally until two jobs are proximate. Then it compares these two jobs according to the standards we stated. We utilize

the CAA algorithm in this comparison. First, it determines whether or not the two tasks are relying on one other. If yes, the impartial task is chosen as the primary resource. If there is no reliance, the CAA compares the total number of new tasks that are dependent on each of these responsibilities. The CAA prioritizes a task with more dependence points than the others. Following that, if both tasks have equal dependence scores, the CAA compares the Number of Requests Ratio (NRR). The assignment with the highest NRR is the most important. If the NRR is equal, the CAA compares the W LR of the neighboring jobs.

- **CAA Algorithm**

1: **Step 1:**

2: Input: $U\ T$ = list of unsorted tasks, n = .()

3: Output: A sorted $U\ T$

4: **Step 2:** Using CAA in $MergeSort$ until $U\ T$ is sorted.

5: **Step 3:** Inside the comparison of Merge sort: u and v are two tasks to compare, P = prioritized task to return from CAA ⊳

Combine sort employs divide and conquer to arrange a list by splitting it into a number of pairs, evaluating them, then merging them again.

6: procedure CAA (u, v)

7: if Duv exists then

8: $P \leftarrow u$

9: else if Dvu exists then

10: $P \leftarrow v$

11: else

12: Compute Du and Dv

13: if $Du > Dv$

Then

14: $P \leftarrow u$

15: else if $Du < Dv$

then

16: $P \leftarrow v$

17: else

18: Compute $NRRu$ and $NRRv$

19: if $NRRu > NRRv$

then

20: $P \leftarrow u$

21: else if $NRRu < NRRv$

then

22: $P \leftarrow v$;

23: else

24: Compute $W\ LRu$ and $W\ LRv$

25: if $W\ LRu > W\ LRv$

then

26: $P \leftarrow v$

27: else

28: $P \leftarrow u$

29: return P

30: End

- **LASA Algorithm**

LASA takes into account four CF characteristics: the number of available processors, Execution Time Ratio (ETR), available RAM, and available bandwidth. We calculate the available total resource against the minimal quantity to get the Processor Ratio (PR), RAM Ratio (RR), and Bandwidth Ratio (BR). This algorithm splits the CF list until only two CFs remain.

Following that, it evaluates the two CFs and performs regression to integrate them until the full list is sorted. It computes the ET R, RR, P R, and BR for each CF to evaluate. The more resources, such as bandwidth, and CPU count, the better. However, the shorter the execution time, the better.

A Linear Attribute Summarized Algorithm

---

1. Input: UF = list of unsorted fog, n = .()
2. Procedure:
3. **Step 1**: Using $LASA$ in $Mergesort$ until UF is sorted
4. **Step 2**: Within the Merge sort comparison: fog A and fog B are two CFs to compare, F = the job with the highest resource to return from LASA Merge sort implements divide and conquer and sorts a queue by partitioning it into a number of pairs, comparing them, and continuing merging.
5. Procedure LASA (UF)
6. Compute $ET\ Ra$ and $ET\ Rb$
7. Compute RRa and RRb
8. Compute $P\ Ra$ and $P\ Rb$
9 Compute BRa and BRb
10 Compute $P\ ointA$ =
11 Compute $P\ ointB$ =
12 if $P\ ointA > P\ ointB$ then
13 $F \leftarrow F\ ogA$
14 else if $P\ ointB > P\ ointA$ then
15 $F \leftarrow F\ ogB$
16: else
17 if $ET\ Rb \geq ET\ Ra$
then

18 $F \leftarrow F\ ogA$

19 else

20 $F \leftarrow F\ ogB$

21 return F

22 End

3.3.6 Conclusions and Contributions

Our contribution to the development of this system consists of many improvements and adjustments to the scheduler's structure in order to get more accurate results to meet the primary aim of resource scheduling in FC. First, we introduced a GUI to the basic system, and then we upgraded the model structure by doing more data filtering and classification with a lightweight module, utilizing the CAA method to prioritize activities. Receiving and scheduling tasks from CFs, regulating and tracking CF resource status, allocating a job to the most qualified CF or cloud, and giving execution results as a response to the relevant CF are among its primary functions. Another significant improvement to the system was the use of the LASA algorithm to sort the nodes and assign suitable devices to the data. Saving system overhead; thus if the smaller work is assigned to the more powerful device, extra resources will remain idle, and if the bigger task is assigned to the less powerful device, system overhead is produced regardless of whether the shortest path search is used. We will need to eliminate this overhead and significantly reduce system latency, as well as cut network usage and power consumption by eliminating delays.

CHAPTER FOUR

SIMULATION RESULTS AND DISCUSION

4.1 INTROUCTION

This chapter has been dedicated towards implementing, displaying, and analyzing experimental findings using the proposed system stated in the previous chapter. The proposed system's implementation was carried out utilizing the iFogSim simulator (Gupta et al., 2017). In this chapter, we also compare the simulation results of our scheduling system and its CAA and LASA algorithms to an implemented original system. The final findings indicate a considerable decrease in energy usage while reaching the best network efficiency, as well as an increase in bandwidth and optimal utilization of other resources when compared to the basic design.

4.2 EXPEREMINTAL SETUP AND SIMULATION PARAMETERS

4.2.1 Experimental Setup

In a simulated environment, the assessment of managing and scheduling the resources of the fog layer was carried out. For our experiments, we used a simulation. To simulate the fog environment, iFogSim was employed. We included fog devices and a fog server by integrating their specific properties in the simulator, such as the number of CF under an MF, MIPS, RAM, and preserving all other current features that one host possesses. The experimental system through which the performance of the suggested method is observed is the varied number of tasks submissions. We designed a GUI for the proposed system; this interface includes the original system as well as

our proposed system. Figure 4.1 shows the block diagram of the proposed smart scheduling system.

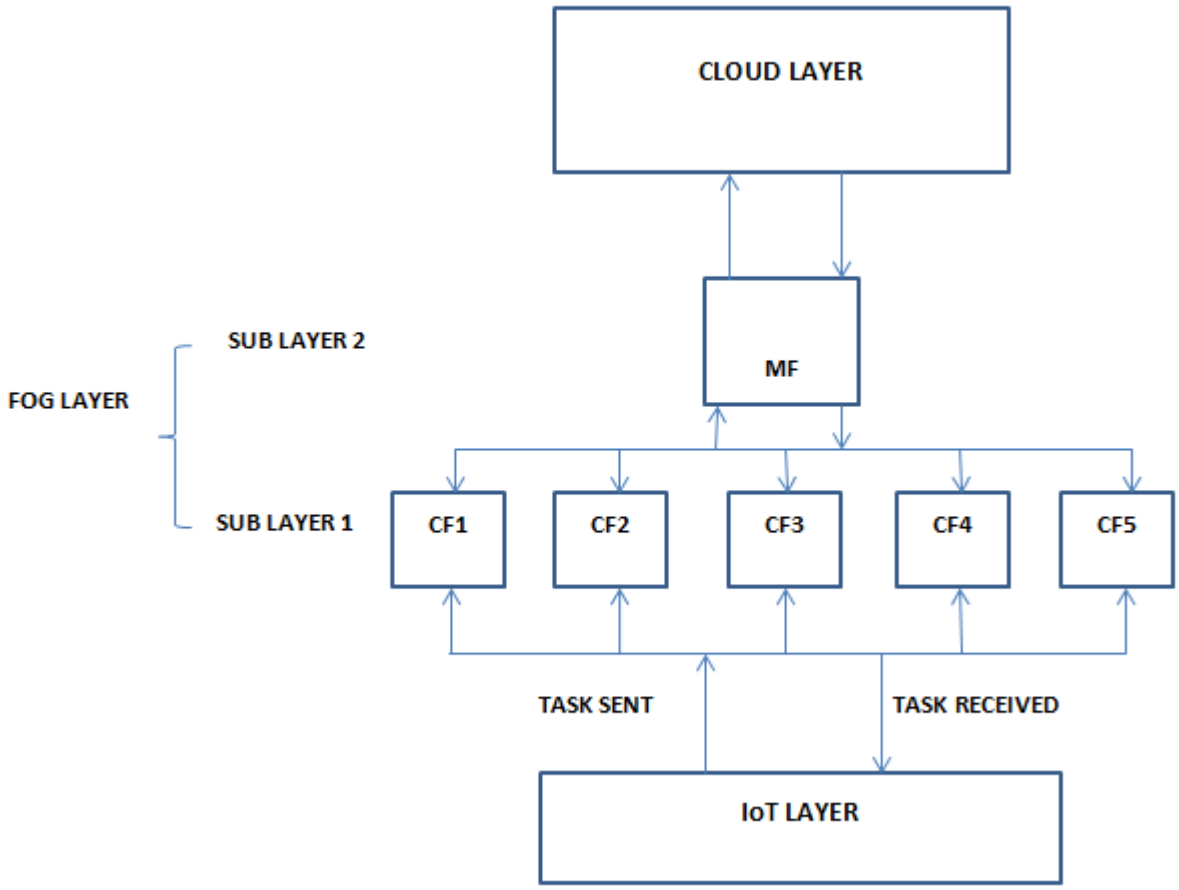

Figure 4. 1: Block Diagram of Proposed System.

Table 4.1 indicated the Simulation Specification, and Table 4.2 indicated the system specification.

Table 4. 2: Simulation Specification

System properties	Specifications
Operating System	Windows 10
Processor	Intel(R) Core(TM)i7-10510U CPU @1.80GHz 2.30 GHz
Storage	RAM 8GB, SSD 256 GB
System Architecture	64 bit, x64
Simulator	iFogSim
Programing Language	java

Table 4. 3: System, CF, and MF resource specification

Attribute	Specification
No. of CF for each MF	5
MIPS	1024 *Random (8,12)
RAM	512* Random (1,3)
UPBW	10000
OWNBW	270
Busy power	82.4333-82.44
Idle Power	1000000
Storage	1024* Random (8,12)
Architecture	X86
OS	Linux
VMM	xen
Time-Zone	10.0
Cost	3.0
Cost per Memory	0.05
Cost per Storage	0.001

Table 4.3 indicated the required resource specification of task for both systems.

Table 4. 3: Required Resource Specification of Tasks

parameters	Specification
Number of task	500 task
RAM	64 MB
MIPS	1024
Task Length	1024KB
Bandwidth	128KB

By utilizing the same parameters for both designs to compare the final outcomes and determine how much of the scheduling modifications we made were effective.

4.2.2 Performance Evaluation

Our proposed smart scheduling architecture comprises three levels: CF, MF, and the Cloud. The framework takes into account each CF's available resources and arranges activities to ensure appropriate resource use. The iFogSim simulator is used to model our scheduling-based optimal resource scheduler architecture. Furthermore, we examine a three-layer framework simulated in the iFogSim simulator to evaluate the efficiency of our system. Peripheral devices, gateway, and the cloud are the tiers. If the MF is unable to provide adequate resources, the job is routed to the cloud. The iFogSim simulator includes a Sensor class that generates Tuple tasks that are processed by FogDevices and received by Actuator. To utilize the FogDevice class as CF, MF, and Cloud, we leverage the Object-Oriented programming feature Inheritance. For both models, we use the same software environment

and CF devices in the simulation. Moreover, iFogSim generates similar tasks. The iFogSim simulator's PowerModelLinear class is used to compute the energy spent by a node. To assess the efficiency of the model we calculated the consumed RAM, energy consumptions, bandwidth, and MIPS for each model and compare the results. We also calculate the simulation time and the execution time of each system.

- **SDFC System**

  The simulation of the basic SDFC system showed the following results. Figure 4.2 shows the simulation console result of the SDFC system.

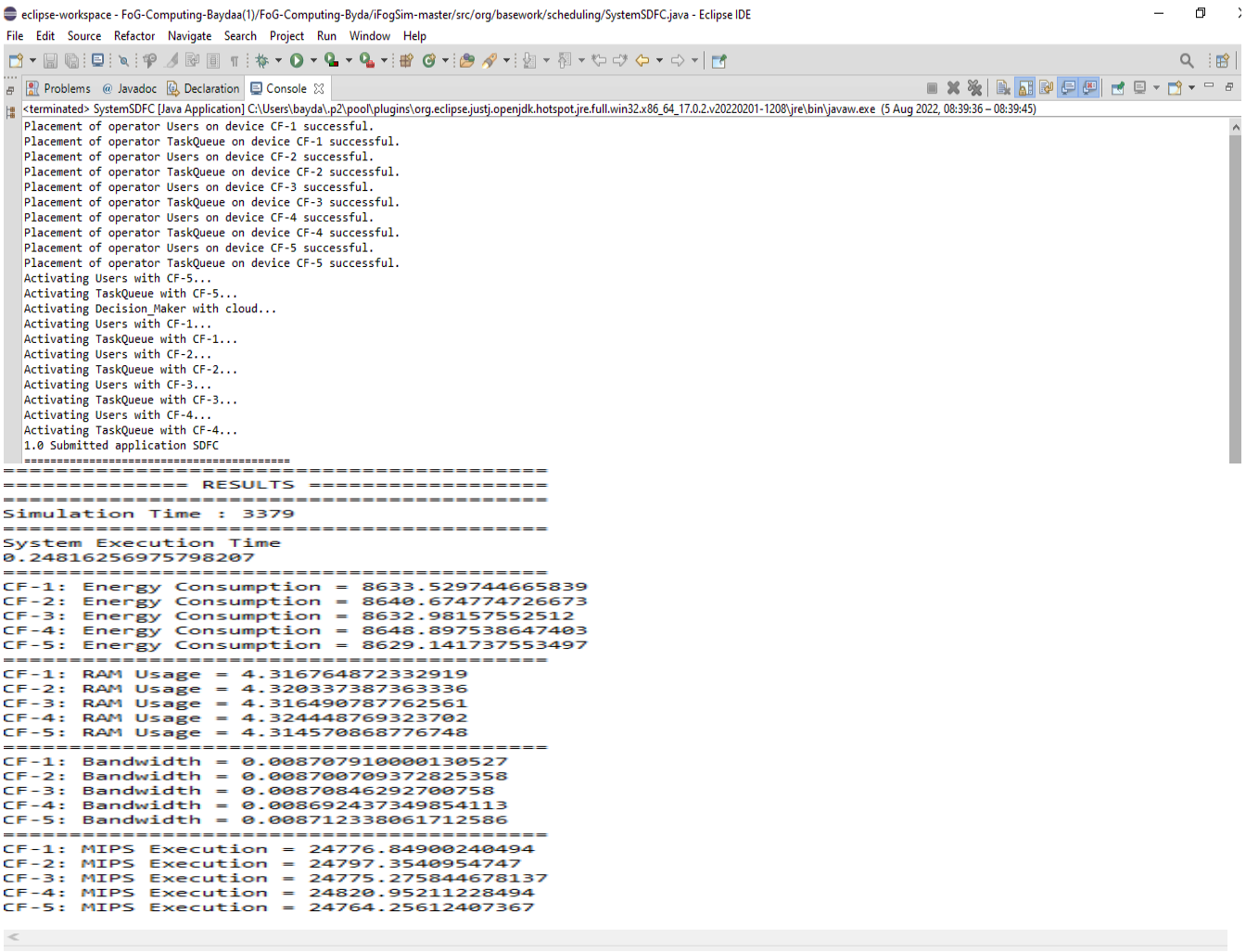

Figure 4. 2: SDFC Console Result

Our study focused on four critical resources, bandwdith, energy consumption, consumed RAM,and CPU processing(MIPS). which are depicted in Figure 4.3.

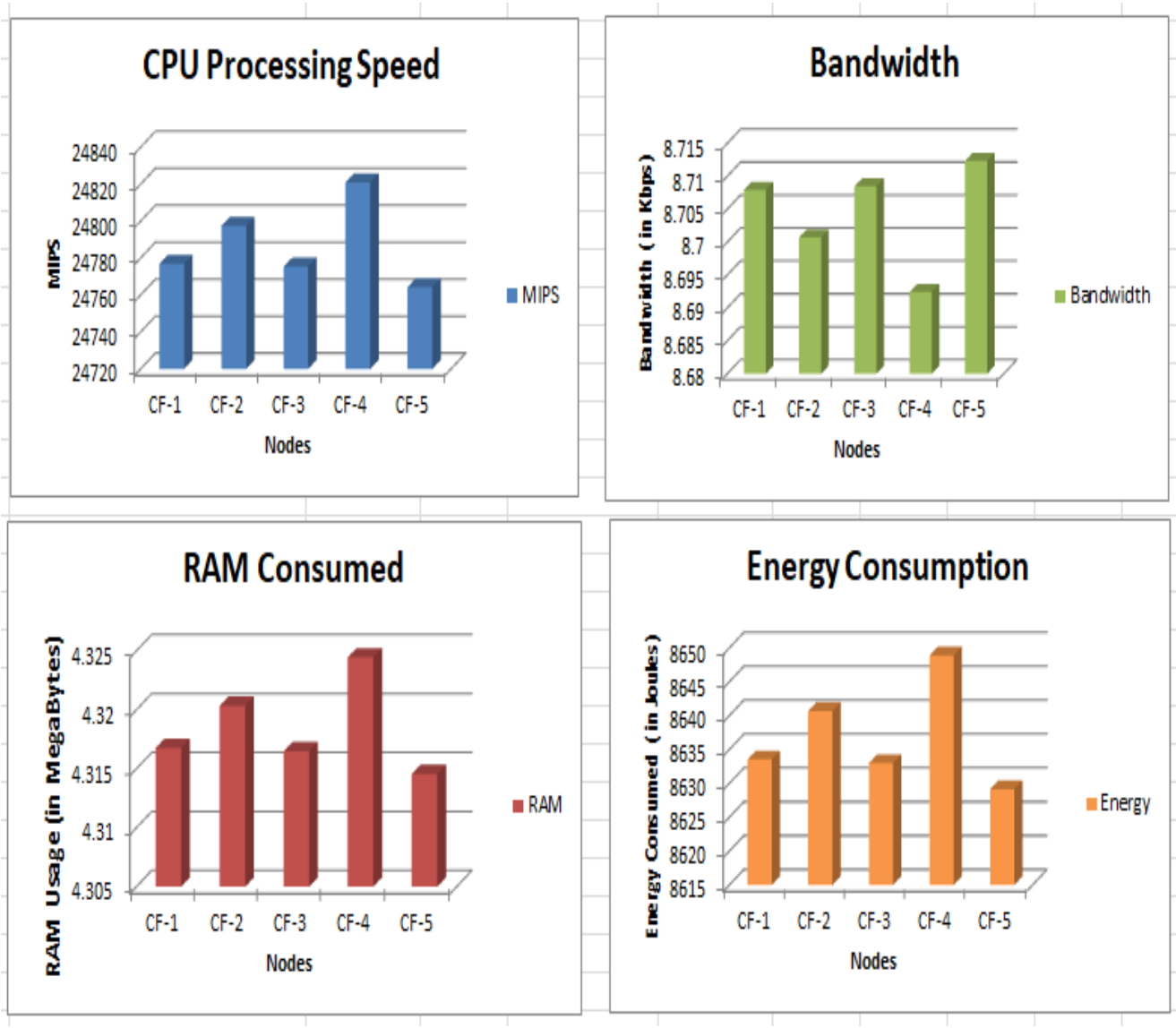

Figure 4.3: Available Resources When Using SDFC System.

- **Proposed System**

  The simulation of our proposed system showed the following results.

  Figure 4.4 shows the simulation console result of the proposed system.

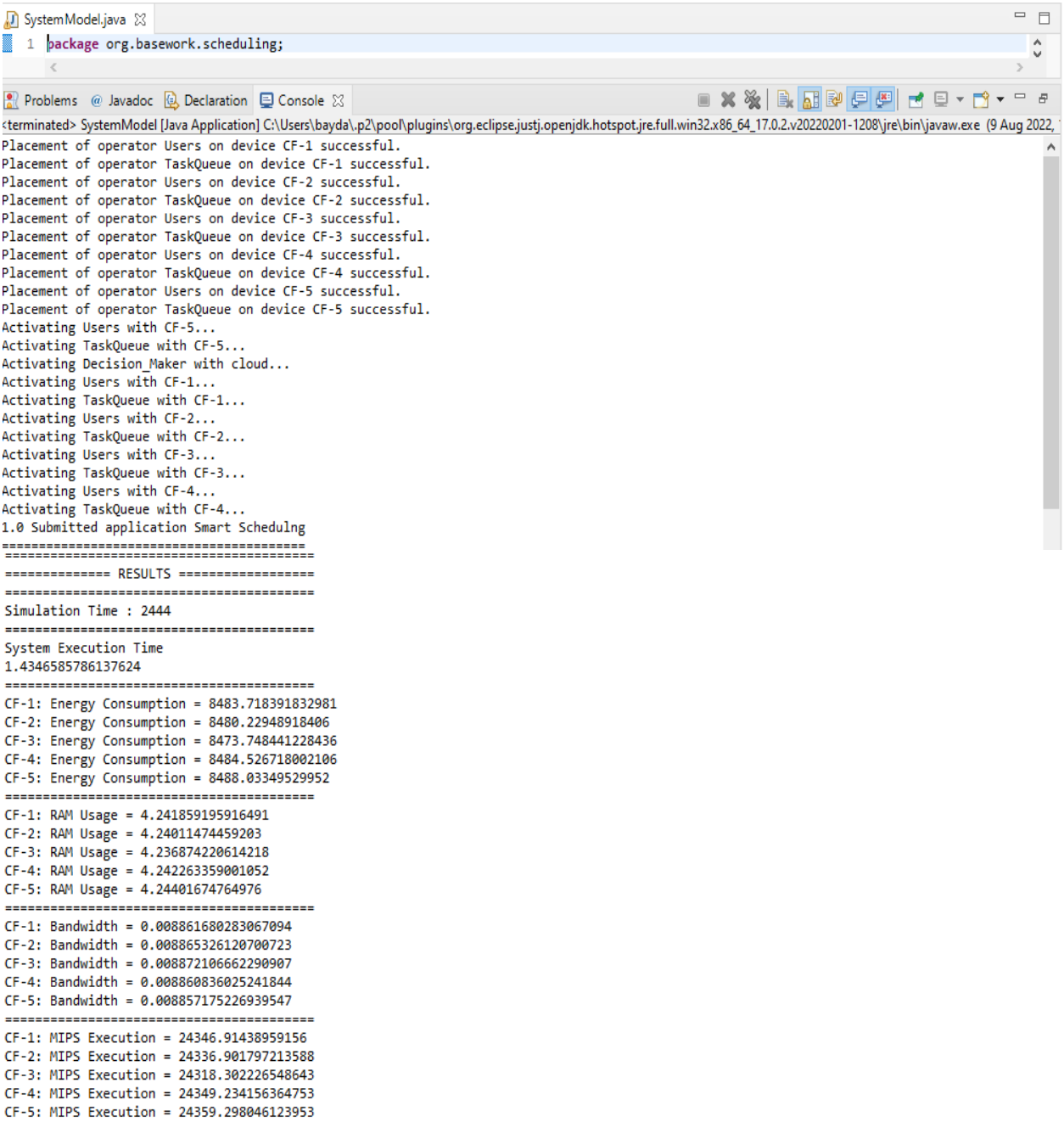

Figure 4.4: Proposed Smart Scheduling Console Result.

Figure 4.5 depicts our research on four essential resources: bandwidth, energy usage, RAM consumption, and MIPS.

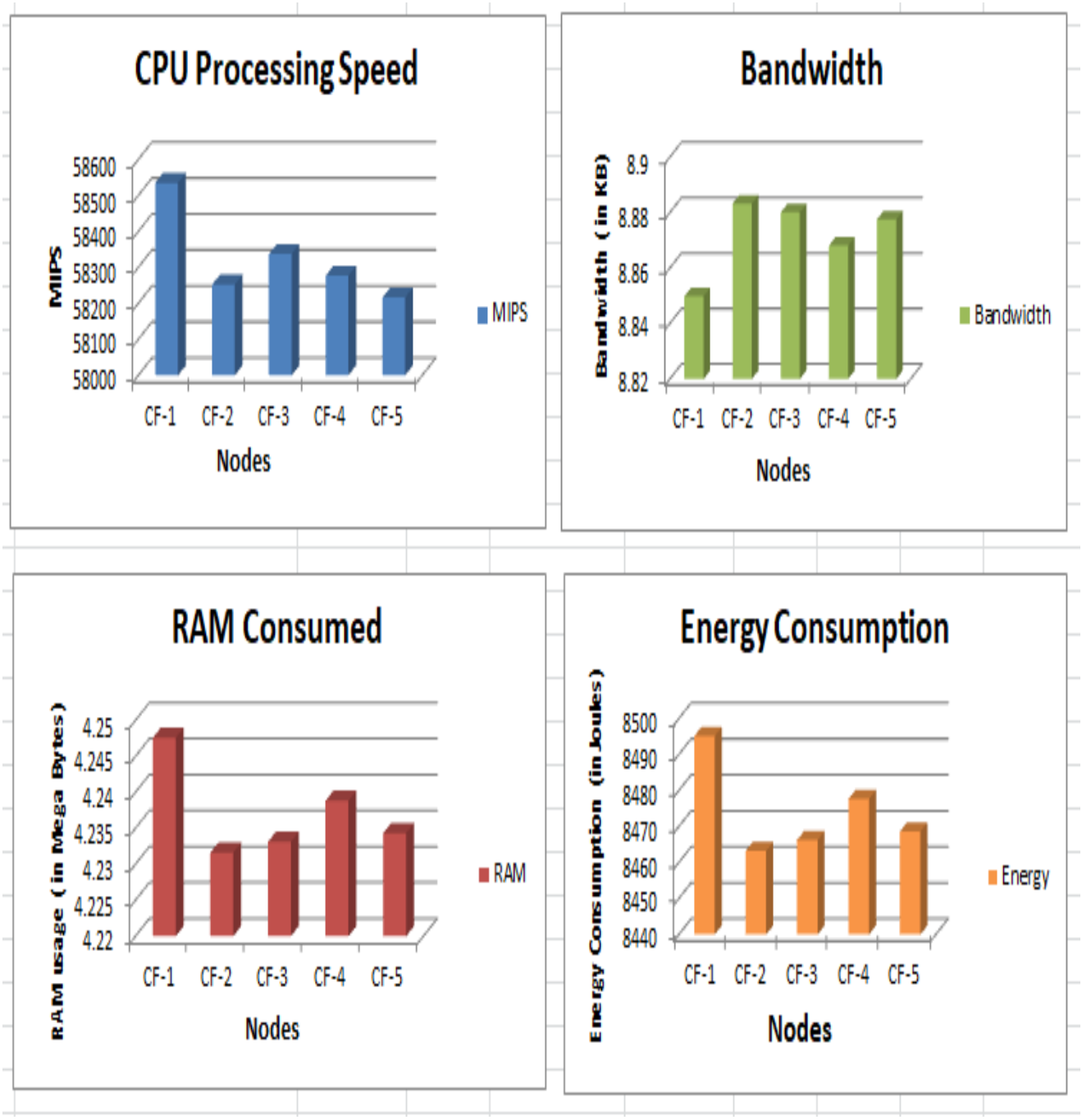

Figure 4. 5: Available Resources When Using Proposed Scheduling System.

4.3 Result and Discussion

To assess the efficiency of our proposed system, we compared the results with the basic system. Evaluating the results under the same conditions and using the same parameters and variables helps us to evaluate the efficiency of the scheduler if we take everything into consideration the stabilization of the number of tasks and resources available for both systems. The proposed framework prioritizes jobs and organizes CF nodes based on resource availability. As a result, once MF receives a job, it instantly sends it to the most qualified CF. In the simulation; we examined 500 tasks assuming 10 users requesting these tasks at the same time. Despite the fact that the same idea and algorithms were employed, the final results demonstrate a significant improvement in the amount of resources utilized and fully exploited in the fog layer. This is due to improvements made to the structure of algorithms, the order of points, and updates applied in scheduled programming. Our contribution to the development of this system consists of many improvements and adjustments to the scheduler's structure in order to get more accurate results in to accomplish the primary aim of resource scheduling in fog computing. First, we introduced a GUI to the basic system, and then we upgraded the system structure by doing more data filtering and classification with a lightweight module, utilizing the CAA method to prioritize activities. Its core roles include receiving and scheduling tasks from CFs, regulating and tracking CF resource status, allocating a task to the most qualified CF or cloud, and giving execution results as a response to the relevant CF. Another significant improvement to the system was the use of the modified LASA algorithm to sort the nodes and assign suitable devices to the data. Saving system overhead, thus if the smaller work is assigned to the more powerful device, extra resources will remain idle, and if the bigger task

is assigned to the less powerful device, system overhead is produced regardless of whether the shortest path search is used. We will need to eliminate this overhead and significantly reduce system latency, as well as reducing network usage and power consumption by eliminating delays.

4.3.1 RAM Usage

The problem of RAM usage is one of the most serious issues that every device faces when doing operations. The CPU and RAM are connected to all computer memory. However, the high memory utilization issue is mostly caused by the overpopulation of several internal processes. This issue causes the network's work to be delayed, reducing efficiency. From the results of the basic system we noticed that memory is very busy when performing tasks. Consequently, it leads to delays in the implementation of the required tasks in addition to an increase in the energy consumed. By using updated algorithms and eliminating looping processes in our proposed system, we were able to significantly reduce the RAM needed, hence speeding up the tasks executions, lowering the execution time. Figure 4.6 shows the comparison between the basic system and the proposed smart system. From the chart below based on the results of the two systems, we note that the proposed smart scheduler reduced the memory used by less than half, achieving speed in performing tasks and reducing delays.

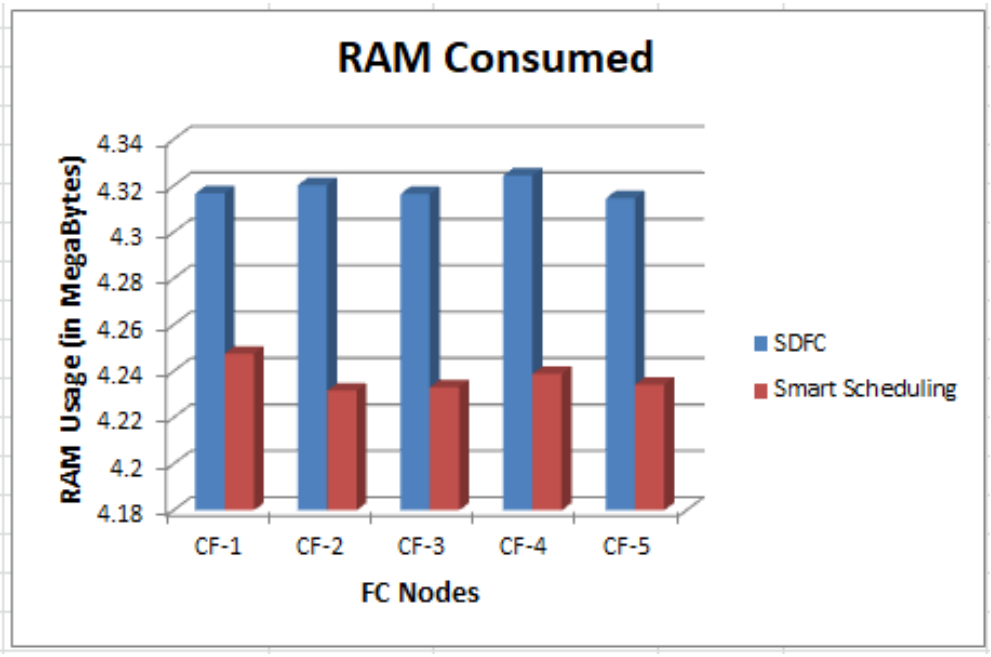

Figure 4.6: Comparison of RAM Consumed by SDFC System and Smart System.

4.3.2 Bandwidth

Bandwidth is defined as complete amounts of data that may be carried from one point to another within a network in a period of time. Bandwidth is commonly depicted as a bit rate and measured in bits per second (bps). The concept bandwidth relates directly to the transmission capacity of a link and is an important aspect of determining network quality and speed. When the results of the two systems were compared, it is evident that the smart scheduler makes the most of the bandwidth available in the fog layer, decreasing the requirement for cloud layer bandwidth. Keep in mind that while many computing activities are performed locally by the fog nodes, only a tiny fraction of the data is transferred to the cloud. Figure 4.7 illustrates that the proposed system provides a greater bandwidth than the basic system.

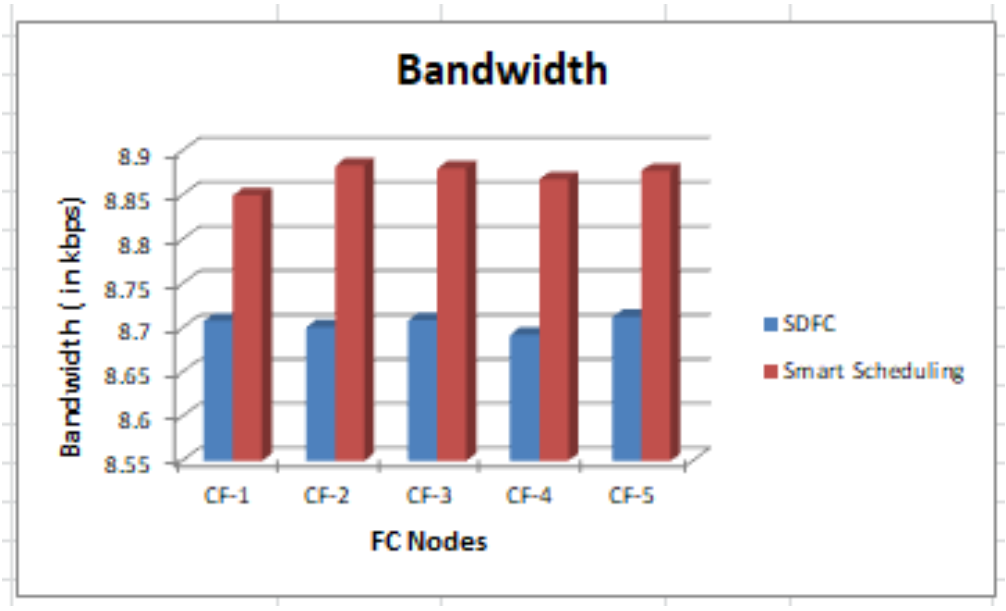

Figure 4.7: Comparison of Bandwidth for SDFC System and Smart System.

4.3.3 MIPS

Million instructions per second (MIPS) is a fair approximation of a computer's actual processing capacity. MIPS is a generic measure of computing performance and, by extension, the amount of work that a bigger machine can do. MIPS is a means to quantify the cost of computing for large servers or mainframes: the more MIPS supplied for the money, the greater the value. According to the final results of our system and compared to the original system, the percentage of our schedulers achieved results more than half of the original system. This high percentage increases the speed of data processing and the speed of response to tasks and thus increases the efficiency of the execution of these tasks. Figure 4.8 illustrates that the proposed smart system has a higher (MIPS) value than the basic system.

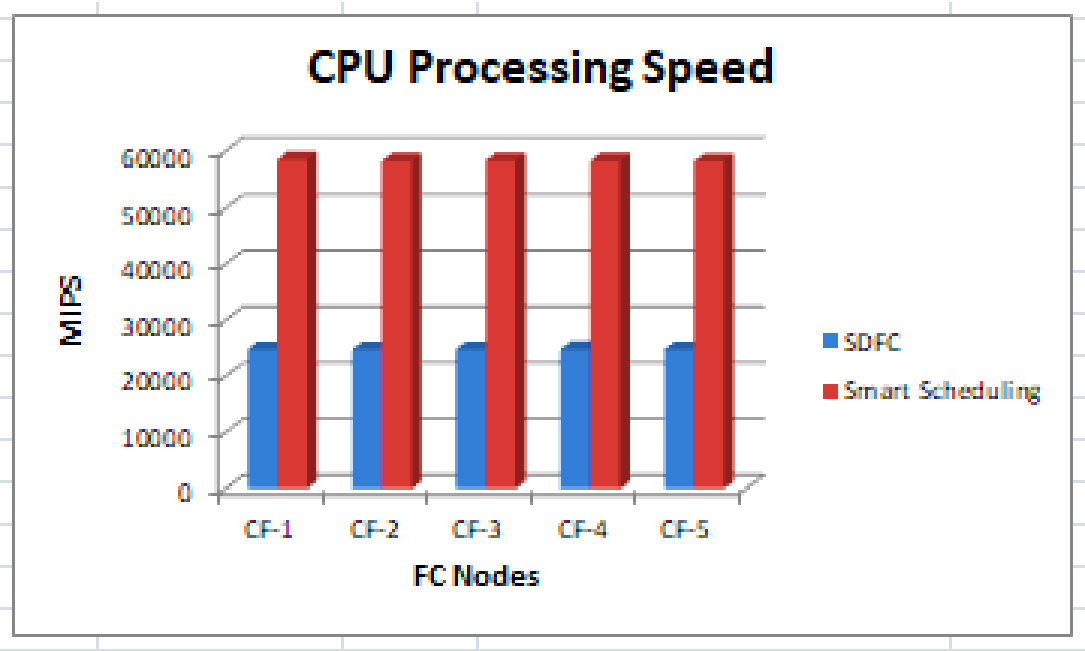

Figure 4.8: Comparison of MIPS Execution for SDFC System and Smart System.

4.3.4 Energy Consumption

Energy consumption is quickly growing, posing a serious economic and environmental concern. At the moment, network infrastructure accounts for a sizable share of the energy footprint. As a result, the notion of energy efficient or green networking has been established. One of the network industry's primary concerns is reducing energy consumption in network infrastructure due to possible economic advantages, social duty, and environmental effect. Using the proposed system of the scheduler and comparing the final results with the original system. Note that the use of this proposed scheduler reduces the energy consumed by half compared to the original system. This improvement is one of the most important additions to the proposed schedule. Figure 4.9 shows the comparison between the two systems, indicating that the percentage of energy consumption in the proposed smart system has decreased by more than half.

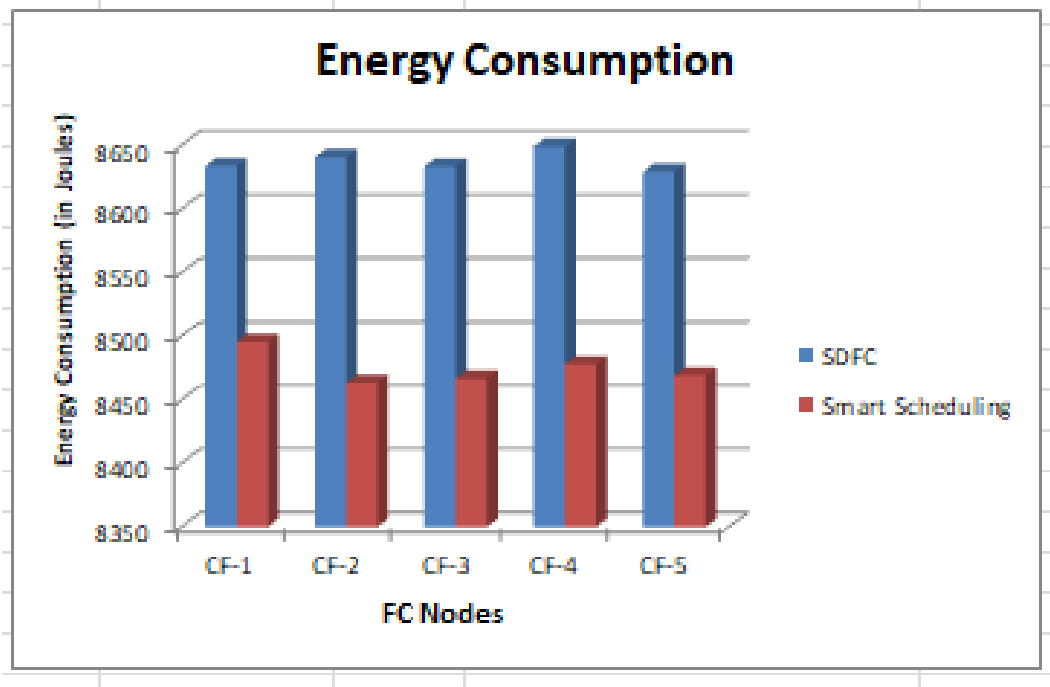

Figure 4.9: Comparison of Energy Consumption for SDFC System and Smart System.

4.3.5 Execution Time

The execution time is described as the time consumed by the system accomplishing that action, including time spent executing run-time or system activities on its behalf. The execution time of an instruction is influenced by how other instructions in the scheduler behave. When two instructions compete for the same resource, they will execute more slowly than if they were performed at widely spaced times. Using the proposed smart system of the smart scheduler, we were able to reduce the time of execution of tasks by a significant percentage as shown in Figure 4.10, where the suggested system's findings were compared to the original system.

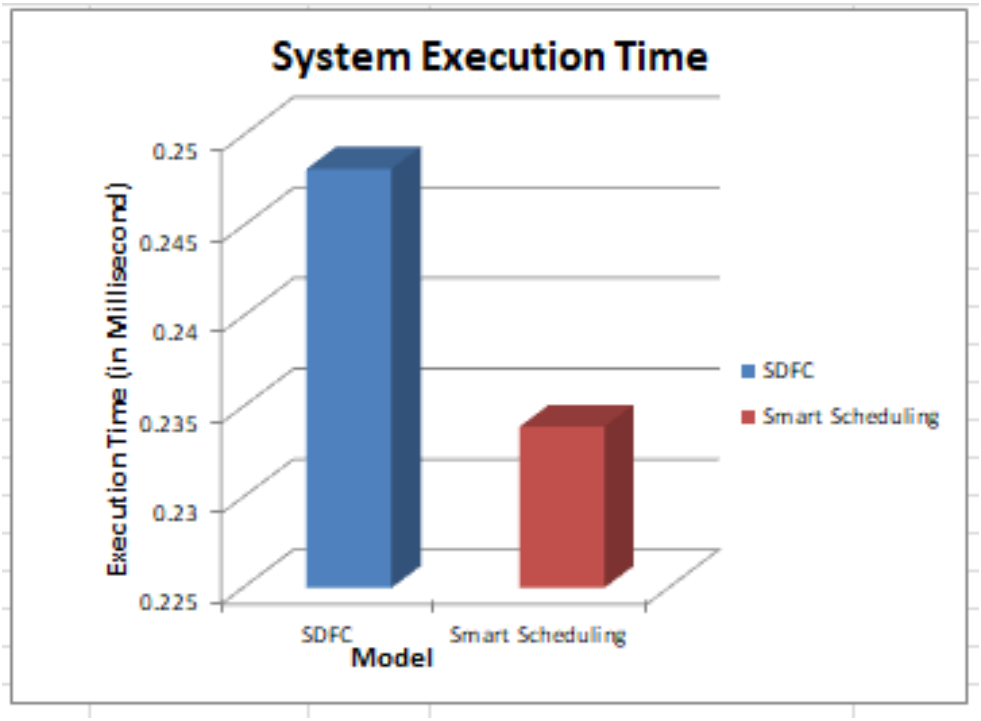

Figure 4.10: Comparison of Execution Time of SDFC System and Smart System.

4.3.6 Accomplished Tasks at Fog Layer

Accomplished task at fog layer reflects the number of tasks completed in the fog layer. By eliminating the cloud layer and calculating the number of tasks completed in the fog layer for the precise number of tasks. It keeps track of how many tasks are accomplished in the fog layer. The final outcomes of both systems are compared; we observe that the presented smart scheduler performs more tasks than the original system. This indicates that the proposed smart scheduler makes the best use of the fog layer's available resources. As a result, the intended aim of using the scheduler to schedule sources has been met. Figure 4.11 shows the comparison between the two systems showing that the number of tasks performed in the fog layer by the proposed smart system is significantly higher than the number of tasks performed by the original SDFC system.

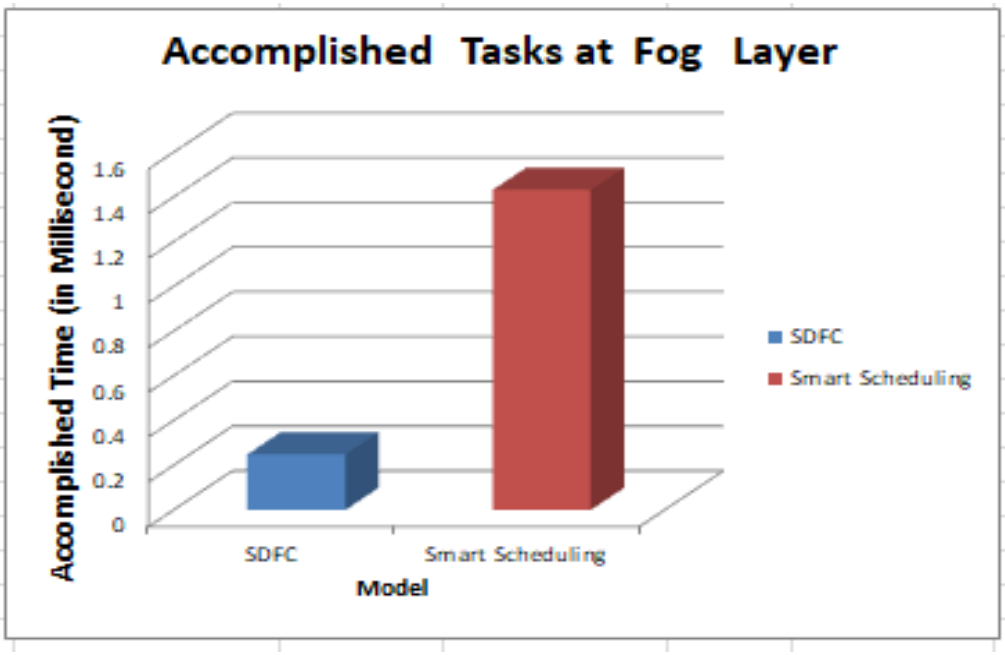

Figure 4. 11: Comparison of Accomplished Tasks at Fog Layer for Both SDFC System and Proposed Smart System.

CHAPTER FIVE

CONCLUTIONS AND FUTURE WORKS

5.1 CONCLUTIONS

In this thesis, a smart resource scheduling system has been presented as a methodology for improving resource consumption in fog computing paradigm. The proposed system is based on the idea of sub-layering of the fog layer. The scheduler architecture includes a master fog layer that sits between citizen fogs, which are general fog nodes, and the Cloud. The master fog is assigned to bear all computing overheads for task scheduling and instantly distributing the highest priority work to the most qualified citizen fog. The LASA algorithm is used by the master fog to prioritize jobs and the CAA algorithm is used by the citizen fogs to sort them depending on their computing power. Furthermore, the scheduler ensures that the citizen fog layer's resources are properly utilized, decreasing reliance on the cloud. To analyze the proposed solution, a simulation environment was required, taking into account the architecture of the fog computing environment. iFogSim was used to create a simulation architecture and environment. 500 tasks were investigated in the simulation. The scheduling approach utilizes CAA and LASA algorithms to ensure that the resources of certain citizen fogs are efficiently handled. Requests sent from users are sent to nearest citizen fog which will handle the request according to available resources. The CAA algorithm will ensure that tasks are scheduled between citizen fogs and in case of not having enough resources it sends the request to the master fog layer which through LASA algorithm will find the best available citizen fog. In the event that the required resources are not available, it will be decided to send the application to the cloud layer. The final comparison results indicate more scheduling of fog layer resources by increasing the number of

processed requests in this layer and reducing the number of processed requests in the cloud layer. There was an increase in available bandwidth as well as an increase in the number of MIPS after employing this scheduler, indicating more utilization of fog layer resources. This increase is compensated by a drop in the level of RAM used, as well as a decrease in the amount of energy consumed to achieve the highest network efficiency and to fulfill the main purpose of employing fog computing.

5.2 SUGGESTIONS FOR FUTURE WORKS

Resource scheduling in fog computing is one of the most important factors that affect network efficiency. This scheduling exploits the sources of the fog layer and prevents the need for the cloud layer. By reviewing the literatures conducted in Chapter Two, we notice the tendency of researchers to work on this challenge. A lot of ideas and systems were proposed in previous years to schedule these resources. Despite this, these proposed solutions are still considered primitive and need a lot of work and improvement to get the maximum scheduling of these sources. The system provided in this thesis is constructed in a way that allows for future expansion. There are several future works that can be completed; here are a few examples:

1. Additional research into the flaws of iFogSim and its algorithms and create some algorithms based on another criterion. The iFogSim algorithms are not intended for use with real-world systems. There is a need to create a dynamic algorithm that will aid in the flexible placement of systems during application execution.
2. As an improvement to the proposed system, we may employ different types of algorithms, such as a priority scheduling algorithm that is more suited for the fog layer, and study the system's reaction.

3. Investigate the deep learning technique for optimizing the processing of resources such as data, information, and expertise. To overcome the cluster scheduling problem, a reinforcement learning strategy might be applied.

8th International Engineering Conference on advances in Computer and Civil Engineering Towards Engineering Innovations and Sustainability, (IEC-2022), Erbil-Iraq

2022 8th International Engineering Conference on Sustainable Technology and Development (IEC) | 978-1-6654-7829-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/IEC54822.2022.9807469

Smart Resource Scheduling Model in Fog Computing

Baydaa Hassan Husain
Department of Information System Engineering
Erbil Polytechnic University
Erbil, Iraq
baydaa.hassan@epu.edu.iq

Shavan Askar
Department of Information System Engineering
Erbil Polytechnic University
Erbil, Iraq
shavan.askar@epu.edu.iq

***Abstract*— Fog computing is among the most significant new concepts in recent technological advancement. It addresses various issues of cloud computing by delivering compute, connectivity, storage, and actual services closer to end devices. Conversely, as systems become more automated, the number of task executions by fog devices grows, necessitating the inclusion of more fog devices. In this paper, we suggest a smart Scheduling framework that enhances the usage of current resources instead of installing more fog sources. It has an extra layer called Master Fog (MF) between each of the cloud and specific fogs termed Citizen Fog (CF). The MF is in a better position to decide on CF and cloud deployment. The Comparative Attributes Algorithm (CAA) is used to prioritize jobs, and a Linear Attribute Summarized Algorithm (LASA) is used to choose the most accessible CF with the greatest computing capabilities. The final findings demonstrate a significant reduction in energy consumption compared to the basic design in order to reach the best network efficiency, as well as important advantages represented in the increase of bandwidth availability and efficient utilization of other sources.**

Keywords*— *fog computing, resource, cloud computing, scheduling, IoT.

I. Introduction

Cloud computing (CC) is gaining traction as a new computing technology in a distributed environment model in the era of the internet. More than 70% of computations will be conducted on cloud servers by 2022 [1]. CC has led to enormous economic advantages across a wide range of businesses by providing reliable, guaranteed network storage services [2] [3]. Meanwhile, with the fast development of smart tools, and other Internet of things (IoT), the physical connection is becoming stronger with time; the number of devices linked to the network is increasing exponentially, and
a considerable number of network edge devices demand fast response time reliability [4]. Extended distance transmission has an influence on real-time performance since the Cloud Data Center (CDC) is placed far away from the end-user, and the CDC processes a huge number of IoT applications, increasing the cloud's burden [5]. CC isn't a panacea for all IoT problems [6] [7]. In many delicate situations, such as medical care and transportation networks, long delays in IoT applications might endanger the lives of a patient or lead to accidents [8]. To address these limitations of CC, Cisco developed a new technology known as Fog Computing (FC) in 2012 [9] [10]. FC is a hyper-virtualized system that links end devices to standard CDC to deliver computing, storage, and networking services [11] [12]. The fog enhances IoT applications by moving computing and infrastructure needs closer to edge devices. As a consequence, services and calculations may operate near to terminals and IoT devices that demand real-time processing capabilities with the least amount of delay feasible [13] [14]. FC architecture is separated into three levels. The first layer is the IoT layer, which covers many categories of devices such as cellphones, automobiles, personal computers, and smart home equipment. This layer may perceive its surroundings and gather data using sensor devices, as well as interact with the FC layer via 3G, 4G, 5G, and Wi-Fi technologies. The FC layer, which contains routers, gateways, workstations, switches, and access points, is the second layer [15]. This layer is capable of calculating, networking, and storing data. Finally, the upper layer is cloud, which consists of cloud servers with great processing power [16]. As a consequence, scheduling, executing, controlling, and guaranteeing data security on these devices is complicated, and fog devices' limited capabilities demand higher computing [10] [17] [18]. The target of resource scheduling is to identify the highest resources for the user in order to achieve the best scheduling results, which include increased network throughput, minimizing processing delays, and enhancing Quality of Service (QoS) [10] [19]. Scheduling promotes distribution of resources and application CPU utilization time to achieve high throughput, high responsiveness, high dependability, good security, reduced energy consumption, and competitive prices [20]. The task scheduler evaluates the devices in use to determine which tasks should be carried out next [21] [15]. FC collects and analyzed information from an end device, such as an IoT device, in order to provide a good

978-1-6654-7829-8/22/$31.00 ©2022 IEEE

REFERENCES

AAZAM, M. & HUH, E.-N. Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, 2015. IEEE, 687-694.

AAZAM, M., ZEADALLY, S & .HARRAS, K. A. J. I. C. M. 2018a. Fog computing architecture, evaluation, and future research directions. 56**,** 46-52.

AAZAM, M., ZEADALLY, S. & HARRAS, K. A. J. I. T. O. I. I. 2018b. Deploying fog computing in industrial internet of things and industry 4.0 . .4682-4674 ,14

ABDULKAREEM, K. H., MOHAMMED, M. A., GUNASEKARAN, S. S., AL-MHIQANI, M. N., MUTLAG, A. A., MOSTAFA, S. A., ALI, N. S. & IBRAHIM, D. A. J. I. A. 2019. A review of fog computing and machine learning: concepts, applications, challenges, and open issues. 7**,** 153123-153140.

ABDULQADIR, H. R., ZEEBAREE, S. R., SHUKUR, H. M., SADEEQ, M. M., SALIM, B. W., SALIH, A. A. & KAK, S. F. J. Q. A. J. 2021. A study of moving from cloud computing to fog computing. 1**,** 60-70.

ABURUKBA, R. O., ALIKARRAR, M., LANDOLSI, T. & EL-FAKIH, K. J. F. G. C. S. 2020. Scheduling Internet of Things requests to minimize latency in hybrid Fog–Cloud computing. 111**,** 539-551.

AHMAD, M. A., PATRA, S. S. & BARIK, R. K. 2020. Energy-efficient resource scheduling in fog computing using SDN framework. *Progress in Computing, Analytics and Networking.* Springer.

AHMED, A., ARKIAN, H., BATTULGA, D., FAHS, A. J., FARHADI, M., GIOUROUKIS, D., GOUGEON, A., GUTIERREZ, F. O., PIERRE, G. & SOUZA JR, P. R. J. A. P. A. 2019. Fog computing applications: Taxonomy and requirements.

AHMED, K. D., ASKAR, S. J. I. J. O. S. & BUSINESS 2021. Deep learning models for cyber security in IoT networks: A review. 5**,** 61-70.

AHMED, M., MUMTAZ, R., ZAIDI, S. M. H., HAFEEZ, M., ZAIDI, S. A. R. & AHMAD, M. J. E .2020 .Distributed fog computing for Internet of Things (IOT) based ambient data processing and analysis. 9**,** 1756.

AI, Y., PENG, M., ZHANG, K. J. D. C. & NETWORKS 2018. Edge computing technologies for Internet of Things: a primer. 4**,** 77-86.

AL-KHAFAJIY, M ,.BAKER, T., AL-LIBAWY, H., WARAICH, A., CHALMERS, C. & ALFANDI, O. Fog computing framework for internet

of things applications. 2018 11th International Conference on Developments in eSystems Engineering (DeSE), 2018. IEEE, 71-77.
AL YAMI, M. & SCHAEFER ,D. Fog computing as a complementary approach to cloud computing. 2019 International Conference on Computer and Information Sciences (ICCIS), 2019. IEEE, 1-5.
ALMOBAIDEEN, W., ALTARAWNEH, M. J. I. J. O. S. & NETWORKS 2020. Fog computing: survey on decoy information technology. 15**,** 111-121.
ATLAM, H. F., WALTERS, R. J., WILLS, G. B. J. B. D. & COMPUTING, C. 2018. Fog computing and the internet of things: A review. 2**,** 10.
BELLAVISTA, P., BERROCAL, J., CORRADI, A., DAS, S. K., FOSCHINI, L., ZANNI, A. J. P & .COMPUTING, M. 2019. A survey on fog computing for the Internet of Things. 52**,** 71-99.
BIAN, S., HUANG, X. & SHAO, Z. Online task scheduling for fog computing with multi-resource fairness. 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall), 2 .019IEEE, 1-5.
BONOMI, F., MILITO, R., ZHU, J. & ADDEPALLI, S. Fog computing and its role in the internet of things. Proceedings of the first edition of the MCC workshop on Mobile cloud computing, 2012. 13-16.
CAIZA, G., SAETEROS, M., OÑATE, W. & GARCIA ,M. V. J. H. 2020. Fog computing at industrial level, architecture, latency, energy, and security: A review. 6**,** e03706.
CHALAPATHI, G. S. S., CHAMOLA, V., VAISH, A., BUYYA, R. J. F. E. C. F. S., PRIVACY, & APPLICATIONS 2021. Industrial internet of things)iiot) applications of edge and fog computing: A review and future directions. 293-325.
DA SILVA, C. A. & DE AQUINO JÚNIOR, G. S. Fog computing in healthcare: a review. 2018 IEEE Symposium on Computers and Communications (ISCC), 2018. IEEE, 1126-1131.
DENG, Y., CHEN, Z., ZHANG, D. & ZHAO, M. J. I. C. 2018. Workload scheduling toward worst-case delay and optimal utility for single-hop Fog-IoT architecture. 12**,** 2164-2173.
GAZORI, P., RAHBARI, D. & NICKRAY, M. J. F. G. C. S. 2020. Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach. 110**,** 1098-1115.
GHANAVATI, S., ABAWAJY, J. H. & IZADI, D. J. I. T. O. S. C. 2020. An energy aware task scheduling model using ant-mating optimization in fog computing environment.
GHOBAEI-ARANI, M., SOURI, A. & RAHMANIAN, A. A. J. J. O. G. C. 2020. Resource management approaches in fog computing: a comprehensive review. 18**,** 1-42.

GILBERT, G. M., NAIMAN, S., KIMARO, H. & BAGILE, B. A critical review of edge and fog computing for smart grid applications. International Conference on Social Implications of Computers in Developing Countries, 2019. Springer, 763-775.

GONZALEZ, N. M., GOYA, W. A., DE FATIMA PEREIRA, R., LANGONA, K., SILVA, E. A., DE BRITO CARVALHO, T. C. M ,.MIERS, C. C., MÅNGS, J.-E. & SEFIDCON, A. Fog computing: Data analytics and cloud distributed processing on the network edges. 2016 35th International Conference of the Chilean Computer Science Society (SCCC), 2016. IEEE, 1-9.

GOPE, P. J. C. & SECURITY .2019 LAAP: Lightweight anonymous authentication protocol for D2D-Aided fog computing paradigm. 86**,** 223-237.

GUPTA, H., VAHID DASTJERDI, A., GHOSH, S. K., BUYYA, R. J. S. P. & EXPERIENCE 2017. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. 47**,** 1275-1296.

HABIBI, P., FARHOUDI, M., KAZEMIAN, S., KHORSANDI, S. & LEON-GARCIA, A. J. I. A. 2020. Fog computing: a comprehensive architectural survey. 8**,** 69105-69133.

HOSSAIN, M. R., WHAIDUZZAMAN, M., BARROS, A., TULY, S. R., MAHI, M. J. N., ROY, S., FIDGE, C., BUYYA, R. J. S. M. P. & THEORY 2021. A scheduling-based dynamic fog computing framework for augmenting resource utilization. 111**,** 102336.

HUSSAIN, M. M., BEG, M .S. J. B. D. & COMPUTING, C. 2019. Fog computing for internet of things (IoT)-aided smart grid architectures. 3**,** 8.

IORGA, M., FELDMAN, L., BARTON, R., MARTIN, M. J., GOREN, N. S. & MAHMOUDI, C. 2018. Fog computing conceptual model.

JAVADZADEH, G. & RAHMANI, A. M. J. W. N. 2020. Fog computing applications in smart cities: A systematic survey. 26**,** 1433-1457.

KISHOR, A., CHAKRABORTY, C., JEBERSON, W. J. I. J. O. E. S. M. & SIMULATION 2021. Intelligent healthcare data segregation using fog computing with internet of things and machine learning. 12**,** 188-194.

KUMAR, M., DUBEY, K. & PANDEY, R. Evolution of emerging computing paradigm cloud to fog: applications, limitations and research challenges. 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2021. IEEE, 257-261.

KUMAR, V., LAGHARI, A. A., KARIM, S., SHAKIR, M. & BROHI, A. A. J. I. J. M. S. C. 2019. Comparison of fog computing & cloud computing. 1**,** 31-41.

LI, G., LIU, Y., WU, J., LIN, D. & ZHAO, S. J. S. 2019a .Methods of resource scheduling based on optimized fuzzy clustering in fog computing. 19**,** 2122.

LI, G., WU, J., LI, J., WANG, K. & YE, T. J. I. T. O. I. I. 2018. Service popularity-based smart resources partitioning for fog computing-enabled industrial Internet of Things. 14**,** 4702-4711.

LI, X., LIU, Y., JI, H., ZHANG, H. & LEUNG, V. C. J. I. A. 2019b. Optimizing resources allocation for fog computing-based internet of things networks. 7**,** 64907-64922.

LIU, Z., ZHANG, J., LI, Y., BAI, L., JI, Y. J. J. O. O .C. & NETWORKING 2018. Joint jobs scheduling and lightpath provisioning in fog computing micro datacenter networks. 10**,** B152-B163.

MATROUK, K. & ALATOUN, K. J. I. J. N. D. C. 2021. Scheduling Algorithms in Fog Computing: A Survey. 9**,** 59-74.

MUKHERJEE, M ,.SHU, L., WANG, D. J. I. C. S. & TUTORIALS 2018. Survey of fog computing: Fundamental, network applications, and research challenges. 20**,** 1826-1857.

MUTLAG, A. A., KHANAPI ABD GHANI, M., MOHAMMED, M. A., MAASHI, M. S., MOHD, O., MOSTAFA, S. A., ABDULKAREEM, K. H., MARQUES, G. & DE LA TORRE DÍEZ, I. J. S. 2020. MAFC: Multi-agent fog computing model for healthcare critical tasks management. 20**,** 1853.

NAHA, R. K., GARG, S. & CHAN, A. J. A. P. A. 2018a. Fog computing architecture: Survey and challenges.

NAHA ,R. K., GARG, S., GEORGAKOPOULOS, D., JAYARAMAN, P. P., GAO, L., XIANG, Y. & RANJAN, R. J. I. A. 2018b. Fog computing: Survey of trends, architectures, requirements, and research directions. 6**,** 47980-48009.

NAJAFIZADEH, A., SALAJEGHEH, A., RAHMANI, A. M & .SAHAFI, A. J. C. C. 2022. Multi-objective Task Scheduling in cloud-fog computing using goal programming approach. 25**,** 141-165.

NGABO, D., WANG, D., IWENDI, C., ANAJEMBA, J. H., AJAO, L. A. & BIAMBA, C. J. E. 2021. Blockchain-based security mechanism for the medical data at fog computing architecture of internet of things. 10**,** 2110.

PEIXOTO, M., GENEZ, T. & BITTENCOURT, L. F. J. I. T. O. S. C. 2021. Hierarchical scheduling mechanisms in multi-level fog computing.

POTU, N., JATOTH, C., PARVATANENI, P. J .C., PRACTICE, C. & EXPERIENCE 2021. Optimizing resource scheduling based on extended particle swarm optimization in fog computing environments. 33**,** e6163.

PRABAVATHY, S., SUNDARAKANTHAM, K., SHALINIE, S. M. J. J. O. C. & NETWORKS 2018. Design of cognitive fog computing for intrusion detection in Internet of Things. 20**,** 291-298.

PUTHAL, D., OBAIDAT, M. S., NANDA, P., PRASAD, M., MOHANTY, S. P. & ZOMAYA, A. Y. J. I. C. M. 2018. Secure and sustainable load balancing of edge data centers in fog computing. 56.65-60 ,

RAJ, P. & PUSHPA, J. 2022. Expounding the edge/fog computing infrastructures for data science. *Research Anthology on Edge Computing Protocols, Applications, and Integration.* IGI Global.

REN, J., ZHANG, D., HE, S., ZHANG, Y. & LI, T. J. A. C. S. 20 .19A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. 52**,** 1-36.

SADEEQ, M. M., ABDULKAREEM, N. M., ZEEBAREE, S. R., AHMED, D. M., SAMI, A. S. & ZEBARI, R. R. J .Q. A. J. 2021. IoT and Cloud computing issues, challenges and opportunities: A review. 1**,** 1-7.

SAMANN, F. E. F., ZEEBAREE, S. R., ASKAR, S. J. J. O. A. S. & TRENDS, T. 2021. IoT provisioning QoS based on cloud and fog computing. 2**,** 29-40.

SAMIE, F., BAUER ,L. & HENKEL, J. J. I. I. O. T. J. 2019. From cloud down to things: An overview of machine learning in internet of things. 6**,** 4921-4934.

SHI, C., REN, Z., YANG, K., CHEN, C., ZHANG, H., XIAO, Y. & HOU, X. Ultra-low latency cloud-fog computing for industrial internet of things. 2018 IEEE Wireless Communications and Networking Conference (WCNC), 2018. IEEE, 1-6.

SINGH, J., SINGH, P., GILL, S. S. J. J. O. P. & COMPUTING, D. 2021. Fog computing: A taxonomy, systematic review, current trends and research challenges. 157**,** 56-85.

SUN, Y., LIN, F. & XU, H. J. W. P. C. 2018. Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II. 102**,** 1369-1385.

TADAPANENI, N. R. J. I. J. O. S. R. & TRENDS, E. 2019. Role of fog computing in the internet of things. 5.

VAQUERO, L. M. & RODERO-MERINO, L. J. A. S. C. C. R. 2014. Finding your way in the fog: Towards a comprehensive definition of fog computing. 44**,** 27-32.

VARGHESE, B., WANG, N., NIKOLOPOULOS, D. S. & BUYYA, R. 2020. Feasibility of fog computing. *Handbook of Integration of Cloud Computing, Cyber Physical Systems and Internet of Things.* Springer.

WANG, S., ZHAO, T. & PANG, S. J. I. A. 2020. Task scheduling algorithm based on improved firework algorithm in fog computing. 8**,** 32385-3.2394

WHAIDUZZAMAN, M., GANI, A. & NAVEED, A. Pefc: Performance enhancement framework for cloudlet in mobile cloud computing. 2014 IEEE International Symposium on Robotics and Manufacturing Automation (ROMA), 2014. IEEE, 224-229.

WU, C.-G. & WANG, L. A deadline-aware estimation of distribution algorithm for resource scheduling in fog computing systems. 2019 IEEE congress on evolutionary computation (CEC), 2019. IEEE, 660-666.

XIAO, K., LIU, K., WANG, J., YANG, Y., FENG, L., CAO, J. & LEE, V. A fog computing paradigm for efficient information services in VANET. 2019 IEEE Wireless Communications and Networking Conference (WCNC), 2019. IEEE, 1-7.

YADAV, A. M., TRIPATHI, K. N. & SHARMA, S. C. J. T. J. O. S. 2022. A bi-objective task scheduling approach in fog computing using hybrid fireworks algorithm. 78**,** 4236-4260.

YE, Y., REN, X., WANG, J., XU, L., GUO, W., HUANG, W. & TIAN, W. J. A. P. A. 2018. A new approach for resource scheduling with deep reinforcement learning.

YIN, L., LUO, J. & LUO, H. J. I. T .O. I. I. 2018. Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. 14**,** 4712-4721.

YOUSEFPOUR, A., FUNG, C., NGUYEN, T., KADIYALA, K., JALALI, F., NIAKANLAHIJI, A., KONG, J. & JUE, J. P. J. J. O. S. A. 201 .9All one needs to know about fog computing and related edge computing paradigms: A complete survey. 98**,** 289-330.

ZHANG, C. J. F. G. C. S. 2020. Design and application of fog computing and Internet of Things service platform for smart city. 112**,** 630-640.

ZHANG, P., ZHOU, M. & FORTINO, G. J. F. G. C. S. 2018. Security and trust issues in fog computing: A survey. 88**,** 16-27.

ZHOU, Y., TIAN, L., LIU, L. & QI, Y. J. I. C. M. 2019. Fog computing enabled future mobile communication networks: A convergence of communication and computing. 57**,** 20-27.

ZOU, Z., JIN, Y., NEVALAINEN, P., HUAN, Y., HEIKKONEN, J. & WESTERLUND, T. Edge and fog computing enabled AI for IoT-an overview. 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2019. IEEE, 51-56.

سیستەمی زیرەکی پێشنیارکراو بۆ بەخشتەکردنی سەرچاوەکانی فۆگ کۆمپیوتینگ

نامەیەکە

پێشکەشی ئەنجومەنی کۆلێژی تەکنیکی ئەندازیاری هەولێر کراوە لە زانکۆی پۆلیتەکنیکی هەولێر وەکو بەشێک لە پێداویستیەکانی بەدەستهێنانی ماستەر لە ئەندازیاری سیستەمی زانیاری.

لەلایەن

بیداء حسن حسین

بەکالۆریۆس لە ئەندازیاری کۆمپیوتەر

بە سەرپەرشتیاری

د. شڤان کمال عسکر

پوختە

فۆگ کۆمپیوتینگ ، یەکێکە لە بیرۆکە داهێنەرە گرنگەکان لە پەرەسەندنی تەکنەلۆجیای ئەم دواییە. چارەسەری جۆرەها کێشەی کۆمپیوتینگ کلاود دەکا بە چارەسەرکردن و ، خەزن کردن و خزمە تگوزاری ڕاستەقینە کە نزیکە کۆتایی بیت بۆ بەکارهێنەر. هەرچۆنێک بێت، زۆربەی کۆمپیوتینگ نۆدس سەرچاوەکانی سنوردارکراون. لە ئەنجامدا، بەبێ بوونی خشتەکردنە کی باش دەبیتە هۆی بەرز بونەوەی قورسایی سودەکانی فۆگ کۆمپیوتینگ.بیرۆکەی بەخشتەکردنی سەرجاوەکان بەکاردیت بۆ دیاری کردنی گونجاوترین سەرچاوە بۆ بەکارهێنەران تاوەکو باشترین ئامانج بەدەست بخەن. زۆربەی لێکۆڵینەوەکانی ئەم دواییە لەسەر فراوانکردنی ژمارەی فۆگ نۆدس بوون بۆ باشترکردنی ئەو سەرچاوانەی بەردەستن . ئەمەش بووە هۆی سەرهەڵدانی کۆمەلەیەک کێشەی تر وەک زیادکردنی تێچوون و بڕی وزەی بەکارهاتوو. بۆیە ئامانجی ئەم تێزە پێشنیارکردنی مۆدێلی خشتەکردنە کە سەرچاوە بەردەستەکان لە چینی فۆگ خشتە بکات و بەپێی ئەرکە پێویستەکان دابەشیان بکات بێ ئەوەی پێویست بێت ئەم نۆدانە زیاد بکرێت. چینێکی زیادەی هەیە، لە ماستەر فۆگ (ئێم ئێف)، لە نێوان کلاود و فۆگ بە ستیزن فۆگ (سی ئێف) ناودەبریت ، ئێم ئێف بەرپرسیارە لە جێبەجێ کردنی ئەرکی سی ئێف و کلاود . لۆگاریتمی خەسڵەتە بەراوردکارەکان (CAA) بەکاردێت بۆ پێشەکیکردنی کارەکان و لۆگاریتمی پوختکراوی تایبەتمەندی هێڵی (LASA) بەکاردێت بۆ هەڵبژاردنی بەردەستترین سی ئێف لەگەڵ بەرزترین سەرچاوە ژمێریارییەکان و بۆ شیکردنەوەی چارەسەرە پێشنیارکراوەکان ئای فۆگسیم (iFogSim) بەکارهاتووە بۆ دروستکردنی تەلارسازی و ژینگەیەکی هاوشێوەسازی بەکار هاتووە. ئەنجامە کۆتاییەکان ئاماژە بە خشتەیەکی بەرچاو دەکەن بۆ سەرچاوە بەردەستەکان کە نیشاندەری زیادی پانی بەندن بە ڕێژەی 14٪، ئەمە جگە لە زیادبوونی خێرایی پرۆسەکە بە ڕێژەی 34٪ لە لایەکی ترەوە، کەمکردنەوە هەبوو لە کە نزیکەی 14٪ بەکار دەهێنرا، وکەمکردنەوەی وزە کە بە نزیکەیی 14٪ بەکار دەهێنرا لە RAM.